

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

NETSKOPE, INC.,

Petitioner,

v.

FORTINET, INC.,

Patent Owner.

IPR2026-00515
U.S. Patent No. 12,126,695

**DECLARATION OF MICHAEL FRANZ IN SUPPORT OF PETITION FOR
INTER PARTES REVIEW OF U.S. PATENT NO. 12,126,695**

TABLE OF CONTENTS

	<u>Page</u>
I. INTRODUCTION AND QUALIFICATIONS	3
A. Education	4
B. Work Experience	4
C. Publications.....	9
D. Curriculum Vitae	9
E. Materials Reviewed	9
II. LEVEL OF ORDINARY SKILL IN THE ART	11
III. OVERVIEW OF THE TECHNOLOGY FOR THE '621 PATENT	12
A. Priority Date of the Claims	12
B. The Alleged Invention	13
C. Prosecution History	16
D. Claim Construction.....	16
IV. TECHNOLOGY LANDSCAPE AT THE TIME OF THE ALLEGED INVENTION.....	16
A. Cloud Computing.....	16
B. Cloud Deployment Configuration Settings	19
C. Infrastructure as Code (IaC)	21
D. Combating Security Threats by Sharing Defensive Information	22
V. INVALIDITY BASED ON OBVIOUSNESS—LEGAL STANDARD	23
VI. GROUND 1: CHERNIN RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS	25
A. Technical Overview of Chernin	25
B. Claims 1, 9, and 17	28
1. Claims 2/10/18	47
2. Claims 3/11/19	49
3. Claims 5/13.....	50

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

5.	Claims 8/16.....	54
VII.	GROUND 2: THE COMBINATION OF CHERNIN AND JEUK RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS	56
A.	Technical Overview of Jeuk	56
B.	Jeuk’s Role in Prosecution History	58
C.	Motivation to Combine Jeuk and Chernin.....	62
D.	Claims 1, 9, and 17	66
4.	Claims 2/10/18	74
5.	Claims 3/11/19	75
6.	Claims 5/13.....	75
6.	Claims 8/16.....	76
VIII.	GROUND 3: THE COMBINATION OF HUDIS AND SHANKS RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS.	77
A.	Technical Overview of Hudis.....	77
B.	Technical Overview of Shanks.....	82
C.	Motivation to Combine Hudis and Shanks.....	86
D.	Claims 1, 9, and 17	92
7.	Claims 2/10/18	121
8.	Claims 3/11/19	123
9.	Claims 5/13.....	124
7.	Claims 8/16.....	129
IX.	CONCLUSION.....	131

I. INTRODUCTION AND QUALIFICATIONS

1. I, Michael Franz, have been retained on behalf of Petitioner Netskope, Inc. (“Petitioner”) as a technical expert in connection with the above-captioned petition for *inter partes* review (“Petition”) of U.S. Patent No. 12,126,695 (the “’695 patent”) (Ex-1001). I have been asked to investigate and provide my opinion regarding the state of the art at the time of the alleged invention and whether claims 1-3, 5, 8-11, 13, and 16-19 of the ’695 (the “Challenged Claims”) would have been obvious to a person of ordinary skill in the art (“POSITA”) in view of the prior art discussed below.

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.

3. I am being compensated for my consulting services, including time spent testifying at any hearing that may be held, at my standard consulting rate of \$800 per hour. 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 post-grant review or the co-pending district court litigation, and I have no other financial interest in this post-grant review.

4. It is my opinion, for the reasons explained in this declaration, that the Challenged Claims would have been obvious to a POSITA at the time of the alleged invention:

Ground	Basis	Reference(s)	Challenged Claims
1	§103	Chernin	1-3, 5, 8-11, 13, and 16-19
2	§103	Chernin and Jeuk	1-3, 5, 8-11, 13, and 16-19
3	§103	Hudis and Shanks	1-3, 5, 8-11, 13, and 16-19

A. Education

5. I completed my undergraduate studies with a Diplomingenieur from the Swiss Federal Institute of Technology in Zurich (“ETH Zurich”) in 1989. In 1994, I obtained my Doctorate of Technical Sciences from ETH Zurich. My dissertation was entitled “Code-Generation On-the-Fly: A Key to Portable Software.”

B. Work Experience

6. I am a tenured Full Professor of Computer Science in the Donald Bren School of Information and Computer Sciences at the University of California, Irvine (“UCI”). I am also, by courtesy, a Full Professor of Electrical Engineering and Computer Science in the Henry Samueli School of Engineering at UCI. In 2016, the University awarded me the title of distinction of “Chancellor’s Professor” and in 2022, I was further elevated to “Distinguished Professor,” the highest campus-level title of distinction at my university. In 2025, I was additionally appointed as an “ICS

Alumni Professor.” This endowed chair, which I hold concurrently with my other appointments, was donated by alumni of our school.

7. I have served as a visiting professor at ETH Zurich, the University of Klagenfurt in Austria, the Technical University of Brunswick in Germany, and the University of Ulm in Germany. I currently hold a courtesy appointment as a visiting professor at the Technical University of Berlin, Germany.

8. 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, which currently stands at more than 530,000 members.

9. I have also been elevated to Fellow of the Association for Computing Machinery (ACM), the global professional society for computer scientists. Fellow is the highest of ACM’s four grades of membership. 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, which currently stands at about 110,000 members. In recent years, ACM has typically elevated no more than 50 individuals to Fellow status in a single year; in 2015, the year I was advanced, there were 42 new Fellows. Furthermore, in July of 2026, I was named one of 80 Inaugural Members of the ACM

SIGSOFT Software Engineering Academy. The Academy honors individuals whose cumulative contributions have shaped software engineering through research, innovation, and service.

10. I have also been elevated to Fellow of the American Association for the Advancement of Science (AAAS), the world's largest general scientific society, with over 120,000 members.

11. Lastly, I am a member of the inaugural cohort of Fellows of the International Federation for Information Processing (IFIP), the global organization for researchers and professionals working in the field of computing. Established in 1960 under the auspices of UNESCO, IFIP is recognized by the United Nations and links some 50 national and international societies and academies of science with a total membership of over half a million professionals. IFIP is an international, non-governmental organization that operates on a non-profit basis.

12. In April of 2021, the Association for Computing Machinery (ACM) named me the recipient of the ACM Charles P. "Chuck" Thacker Breakthrough in Computing Award. I am being recognized for the development of just-in-time compilation techniques that enable fast and feature-rich web services on the internet. This is the third highest award across all of computer science given by a professional

society, behind only the Turing Award (“the Nobel Prize of Computing”) and the ACM Prize in Computing (for early to mid-career contributions).

13. Furthermore, I am a recipient of the IEEE Computer Society’s Technical Achievement Award, “for pioneering contributions to just-in-time compilation and optimization and significantly advancing Web application technology.” At most 4 of these awards are given annually by the IEEE Computer Society, the largest of the IEEE’s technical societies with a current membership of more than 375,000 members. I am also a recipient of the Humboldt Research Award, also known as the “Humboldt Prize.” The award, given by the Alexander von Humboldt Foundation of Germany and funded by the German federal government, recognizes renowned researchers outside of Germany whose “fundamental discoveries, new theories or insights have had a significant impact on their own discipline and who are expected to continue producing cutting-edge achievements in the future.” It is the highest award given by the Foundation to researchers based outside of Germany. In 2025, I was awarded an “ERC Advanced Grant” by the European Research Council, the highest research award given to individual researchers by the European Union.

14. I have led pioneering research on downloadable code in client-server settings such as what we today call “Web 2.0.” This research has had a real and lasting impact on a great many people. I am the co-inventor (with one of my former Ph.D.

students) of the “Trace Tree” compilation technique, for which the United States Patent and Trademark Office has awarded U.S. Patent No. 8,769,511. I collaborated with the non-profit Mozilla Foundation to incorporate this technique into the Firefox web browser, where it became the basis of the “TraceMonkey” JavaScript engine, eventually used by several hundred million people every day.

15. Over the course of my career so far, I have been the Principal Investigator on several high-profile research projects with a total budget of well over \$24M. My expertise in software systems with distinct emphases on performance and security of client-server and mobile computing has been sought out repeatedly by the Federal Government, and I have been participating in many high-level invitation-only meetings on Critical Infrastructure Protection and Cyber Security organized by the National Intelligence Community, the Department of Defense, the Department of Homeland Security, and the Department of Energy. The overwhelming majority of the federally-funded projects that I have worked on over the course of my career have been related to computer security.

16. Before I reached the maximum allowed number of consecutive terms on the editorial board, I was an Associate Editor of one of the flagship journals of the IEEE, the IEEE Transactions on Dependable and Secure Computing (TDSC). I have also served on the editorial boards of two further peer-reviewed scholarly journals

focusing on software engineering, Software Practice and Experience (SPE) and Computer Science Research and Development (CSRD). I have served on the program committees of most major academic conferences that are related to the various themes of my research. I have served as the primary advisor to 41 completed Ph.Ds., served on the dissertation committees of a further 17 graduated Ph.Ds., and currently serve as the primary advisor on 7 further dissertations in progress.

C. Publications

17. In addition to my dissertation, I co-authored “Automated Software Diversity,” released in 2015.

18. I have published 38 reviewed journal and magazine articles since 1993, and more than 120 conference and workshop papers.

19. I am an inventor on six issued U.S. patents.

D. Curriculum Vitae

20. A copy of my curriculum vitae is attached as Ex-1004 to the Petition.

E. Materials Reviewed

21. In forming my opinions, I have reviewed the ’695 patent and its prosecution history, the prior art references discussed below, and the additional exhibits cited throughout this declaration. I have considered these materials in their entirety, even if only portions are discussed here. I have also relied upon my own knowledge, training, and decades of experience in the fields of computer science, software

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

systems, and network security. The following is an exemplary list of the materials on which I based my opinion.

Exhibit No.	Description
1001	United States Patent No. 12,126,695 (“the ’695 Patent”)
1002	Prosecution History of U.S. Patent No. 12,126,695 (“the ’695 Prosecution History”)
1003	Declaration of Dr. Michael Franz
1004	Curriculum Vitae of Dr. Michael Franz
1005	Prosecution History of U.S. Patent No. 11,785,104 (“the ’104 Prosecution History”)
1006	US 2016/0366174A1 (“Chernin”)
1007	US 2019/0123973A1 (“Jeuk”)
1008	US 2017/0207980A1 (“Hudis”)
1009	US 2018/0063085A1 (“Shanks”)
1010	Cloud Computing: Benefits and Risks of Moving Federal IT into the Cloud, July 01, 2010
1011	Cloud Computing Adoption Continues Accelerating In the Enterprise, November 22, 2014
1012	Fortinet and Palo Alto Networks Co-Found the Industrie’s First Cyber Defense Consortium, May 30, 2014
1013	The Mitre Corporation, STIX Language_ Version 1.0 (Official), 2012
1014	The Mitre Corporation, TAXII Services Specification, Version 1.0 (draft) by Mark Davidson and Charles Schmidt, Nov. 16, 2012

Exhibit No.	Description
1015	NIST Special Publication 800-150, Guide to Cyber Threat Information Sharing, 2016
1016	Ex. D to 2026-01-22 Fortinet's Disclosure of Infringement Contentions
1017	OASIS Advances Automated Cyber Threat Intelligence Sharing with STIX, TAXII, CybOX, July 16, 2015

II. LEVEL OF ORDINARY SKILL IN THE ART

22. 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.

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

24. 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 pace of innovation; (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.

25. In my opinion, a POSITA at the time of the alleged invention would have had at least a bachelor's degree in computer science, computer engineering, electrical engineering, or a related field, and at least two years of work or research experience in cloud and distributing computing with exposure to security issues. Work experience can substitute for formal education and vice versa. My opinions in this declaration are offered from the perspective of such a POSITA, and I am qualified to opine as a POSITA given my own education, training, and decades of experience in these fields.

26. Well before November 27, 2017, my level of skill in the art was at least that of a POSITA. My analysis and conclusions herein are from the perspective of a POSITA as of that date. A different priority date would not change my opinions, as all of the prior art relied upon in this declaration was published well before any potential priority date.

III. OVERVIEW OF THE TECHNOLOGY FOR THE '621 PATENT

A. Priority Date of the Claims

27. 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 sufficiently disclosed through its written description the invention claimed in the later-filed U.S. patent application.

28. 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 in an application for patent published or deemed published and that was effectively filed before the effective filing date of the claimed invention.

29. I have not been asked to review the priority chain of the '695 patent. For purposes of this declaration and my opinions, I assume the claims are entitled to the earliest available priority date, which is November 27, 2017.

B. The Alleged Invention

30. The '695 patent has a lengthy specification that is a continuation-in-part of an earlier chain of provisional and non-provisional patent applications. A substantial proportion of the specification is directed to matters outside the direct scope of the Challenged Claims. A much more limited portion of the specification relates to claim-relevant concepts such as “configuration settings.” For example, columns 69-84 and Figures 5-12 are most relevant to the Challenged Claims.

31. The claims of the '695 patent are, generally speaking, directed toward a method by which one cloud deployment learns from another to combat security threats. For example, one cloud deployment can receive configuration settings used

in other cloud deployments to combat security threats, then beneficially adopt one or more of those settings for the purpose of combating security threats. Among such configuration settings to adopt are blacklists of IP addresses, blacklists of network domains, and configuration settings to close security vulnerabilities.

32. Figure 10 (below) depicts a process flow that relates to what is recited in the Challenged Claims.

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

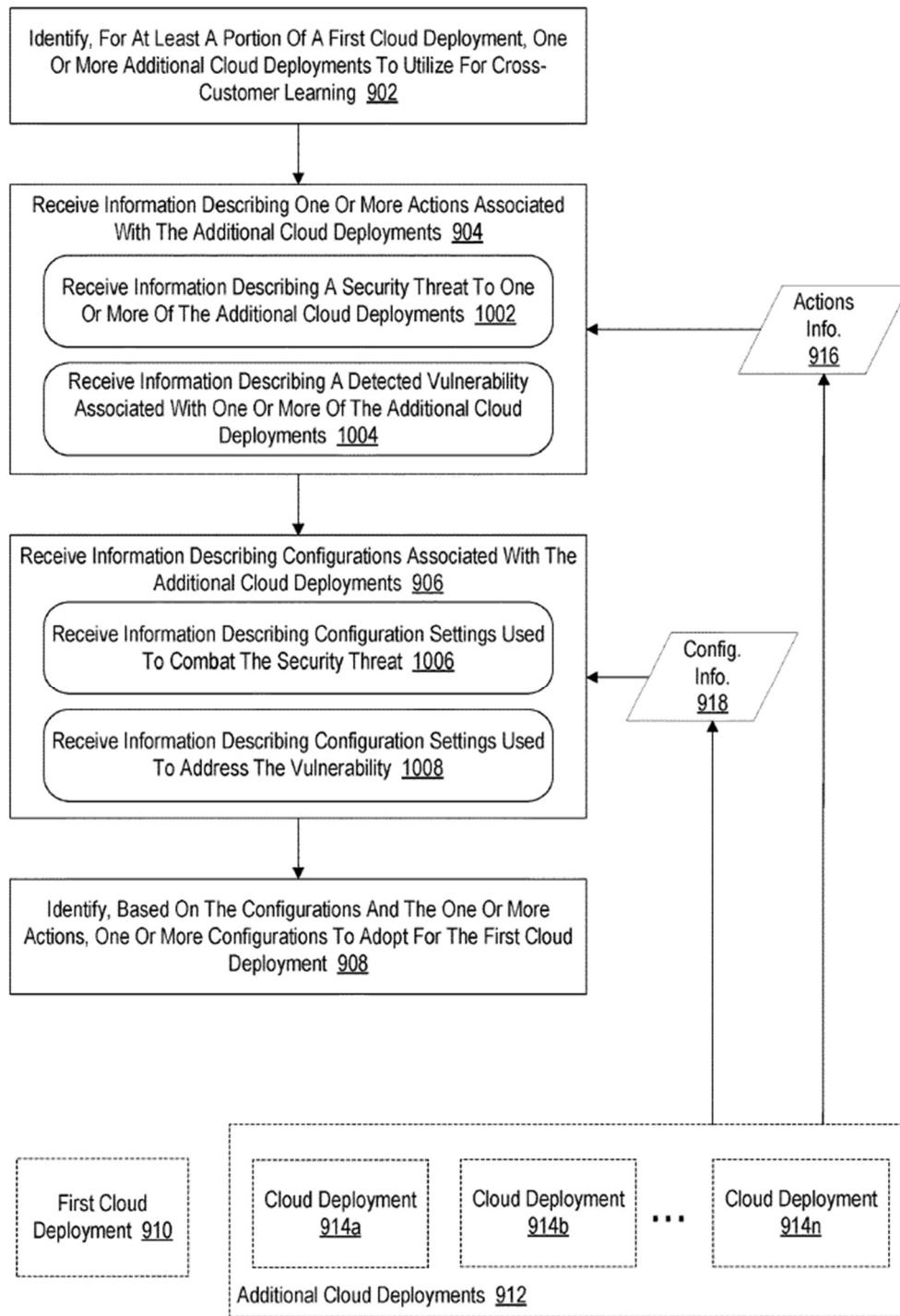


Fig. 10

Ex. 1001, Fig. 10.

C. Prosecution History

33. The claims went through prosecution without an office action. The examiner only “requested a terminal disclaimer be filed to overcome any potential double patenting rejection over US Patent No. 11,785,104”, and the Applicant filed a terminal disclaimer accordingly.

D. Claim Construction

34. I have been instructed that claim terms in an IPR are construed according to the “plain and ordinary” meaning standard of *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) and 37 C.F.R. § 42.100(b). I understand this to mean that the claim terms are interpreted as a POSITA would have understood the claim terms according to their plain and ordinary meaning as informed by the specification and prosecution history. I do not see the need to construe any claim term outside its plain and ordinary meaning.

IV. TECHNOLOGY LANDSCAPE AT THE TIME OF THE ALLEGED INVENTION

35. In this section, I summarize the state of the art at the time of the alleged invention, which in my opinion establishes that each individual element recited in the Challenged Claims was well known, and that these elements were already being combined in practice, well before the earliest priority date of the Challenged Claims.

A. Cloud Computing

36. Cloud computing generally refers to the on-demand delivery of computing resources and services over a network, including the on-demand provision of servers, storage, databases, networking, and software.

37. As the National Institute of Standards and Technology recognized in 2010, the concept of cloud computing was decades-old at the time and had grown into a well-established industry by 2010:

Although the power of modern cloud computing systems is new, the ideas behind cloud computing reach back through decades. In the early 1960s, researchers proposed the idea of computing as a utility, similar to other services such as gas or electricity. Around the same time, techniques to make a single computer appear to be many separate “virtual” computers were developed and implemented on mainframe computers. Some of the building blocks for cloud computing were in place, but performance and costs were barriers, and networking was inadequate. Years of hardware advances were needed to close the gap. By the 1990s, the Internet had made grid computing possible: many computers working together on a single problem over a network. By the 2000s, the term cloud computing was being used to describe computing services delivered over a network, and, in 2010, a substantial and growing number of vendors are developing cloud computing offerings for government, industry, and the general public.

Ex. 1010, pg. 2.

38. In 2014, cloud computing adoption continued to accelerate, with a significant number of enterprises using cloud computing. For example, at that time, an IDG

study found that “69% of enterprises have either applications or infrastructure running in the cloud....” Ex. 1011, pg. 1.

39. Accordingly, by the time of the alleged invention (November 2017), not only was this concept of “cloud computing” well established and widely known, even to lay persons, but the original notion of cloud computing had further differentiated into a range of cloud service models that all have names ending in “as a service.” In 2017, these included:

- Infrastructure as a Service (IaaS): virtual servers, storage, and networking provided on demand. This eliminates the need to purchase and maintain physical data centers, allowing customers to scale resources on demand and pay only for what they use. Examples are Amazon Web Services (AWS), Google Cloud, and Microsoft Azure.
- Platform as a Service (PaaS): managed runtimes and application platforms. This allows developers to build, test, and deploy applications without having to manage the underlying services or infrastructure. Examples are platforms such as Google AppEngine and Microsoft Azure App Services.

- Software as a Service (SaaS): complete software applications hosted by a provider and delivered over the web. Examples are applications such as Microsoft's Office 365.

40. These cloud service models in turn can be deployed on public clouds such as Amazon's AWS or Microsoft's Azure, on private clouds in which all software and hardware resources are dedicated to a single tenant, or on hybrid clouds combining the two. Private clouds in turn can be hosted either on premises inside the tenant's organization, at a third party co-location facility, or managed off-site by a vendor. A POSITA would have known the most important cloud service models, would have known some of the most important providers of these services, and would have understood the distinction between public, private, and hybrid clouds by November 2017.

41. Additional elements of computing infrastructure could be turned into further "as a service" models. For example, a POSITA who had never been exposed to the concept of "database as a service," but who knew about databases, would instantly have recognized that databases could also be offered as a cloud service model in a manner analogous to IaaS, PaaS, and SaaS.

B. Cloud Deployment Configuration Settings

42. To be commercially viable across a wide range of potential users, cloud offerings (including those available in 2017) are typically designed to be highly flexible, and as a result, they typically have many user-configurable parameters. Such a set of parameters that customizes a cloud deployment for its intended use is often called a configuration setting, and a file that contains these settings is also called a configuration file.

43. Two distinct ways of specifying a cloud configuration are imperative declaration, in which the configuration includes a step-by-step sequence of commands that need to be executed in sequence to arrive at the desired end state, and declarative configuration, in which the desired end state is described statically (e.g., “I need three web servers and one database”) and the cloud platform or its associated tools automatically determine how to obtain the end result. By the time of the alleged invention, declarative configuration was the dominant paradigm. For example, HashiCorp’s TerraForm product (cited explicitly in the ’695 Patent’s specification at 60:36-39) was a widely known and widely used provider-agnostic tool for managing such cloud deployment configuration settings in 2017.

44. At the time of the alleged invention, cloud deployment configurations could be either imperative or declarative, and, in particular, that configuration settings could be expressed as declarative specifications addressing, among other parameters,

computing resources, networking, storage, and application environments. Once created, such configuration settings could then be used again and again to achieve reliable, secure, and repeatable cloud deployments across one or more “as a service” environments.

C. Infrastructure as Code (IaC)

45. In the declarative approach to cloud configuration, both now and in 2017, a developer specifies the desired end result of the cloud configuration rather than the steps required to get there, and automated tooling does the rest. The declarative specification is essentially a program in a cloud configuration language and enables developers to treat cloud infrastructure as if they were software.

46. This has led to the term “Infrastructure as Code” or “IaC.” The term originated in the context of Infrastructure as a Service (IaaS) to specify the provisioning of virtual servers, networks, storage etc. An early leader in this domain was the TerraForm product by HashiCorp (mentioned above), which had its initial open-source release in July of 2014. However, by the time of the alleged invention and still today, the term IaC had expanded beyond just IaaS to encompass the declarative configuration of deployments for different kinds of cloud service models, including PaaS and SaaS.

47. A POSITA would have understood that a declarative configuration setting for a cloud deployment is “just a program in a special language.” Its power stems from the fact that in order to make an adjustment to a deployed cloud, a developer only needs to change parts of a declarative description in the specification, and the tools will automatically determine the steps required to migrate the actual deployment to the new state. For all practical purposes then, the underlying physical infrastructure becomes (almost) as malleable as software, i.e., it becomes “code.”

D. Combating Security Threats by Sharing Defensive Information

48. At the time of the alleged invention, cyber threats were an important problem, and significant resources were being devoted to efforts of mitigating and eliminating such threats. Cloud deployments would often be particularly attractive targets to an attacker, since a vulnerability in a shared resource might have the effect that everyone using that resource would also be vulnerable as a result.

49. Defensive approaches that used shared threat intelligence were well-known before 2017, such as using information shared among participants of a threat intelligence cloud. For example, in 2014, a group of corporations in the cyber defense industry (Fortinet, MacAfee, Palo Alto Networks, and Symantec) formed the Cyber Threat Alliance, an organization with the goal of “deep collaboration on threat intelligence and sharing of prevention measures.” Ex. 1012, pg. 1. For

example, avoiding configuration settings that had led to exploitation of others in the past, and adopting configuration settings that had resisted exploitation attempts in the past, would naturally and obviously help to improve the defensive posture of multiple participants in a threat intelligence sharing network.

50. Standards such as STIX (Structured Threat Information eXpression), a standardized, machine-readable language used to describe and structure cyber threat information, and TAXII (Trusted Automated eXchange of Intelligence Information), an application-layer protocol used to securely transport and share STIX data over HTTPS using REST APIs were developed around the same time; the original specification drafts were both released by MITRE in 2012. Exs. 1013-1014 and 1017. By 2015, the STIX and TAXII standards were being advanced through the international open standards process at OASIS in collaboration with companies like Boeing, Check Point, Cisco, Dell, EMC, eSentire, Fortinet, Fujitsu, IBM, iboss, iSIGHT Partners, NEC, New Context, Palo Alto Networks, Resilient, Securonix, Soltra, TELUS, ThreatQuotient, ThreatStream, TruSTAR, US DHS Office of Cybersecurity and Communications, US NIST, ViaSat.

V. INVALIDITY BASED ON OBVIOUSNESS—LEGAL STANDARD

51. As discussed above, I have been informed of the legal standards governing patent invalidity based on prior art. I further 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.

52. 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 POSITA at the time the alleged invention was made (and not from the perspective of either a layman or a genius in that art).

53. I understand that obviousness may also be shown by a combination of prior art references where a POSITA would have had a motivation to combine the references, with a reasonable expectation of success. I have applied these standards, together with the ordinary meaning of the claim terms as would be understood by a POSITA and as informed by the specification and prosecution history, in forming the opinions below.

54. I understand that the question of obviousness is to be determined based on: (1) the scope and content of the prior art; (2) the differences between the subject matter of the claim and the prior art; (3) the level of ordinary skill in the art at the time of the alleged invention; and (4) any relevant objective factors (the “secondary indicia”) indicating nonobviousness—including commercial success, a long-felt need, failed attempts by others, copying, unexpected results, praise by others in the field, the

taking of licenses, and expressions of surprise by those skilled in the art—or indicating obviousness, such as independent invention by others before or at about the same time. I reserve the right to address any secondary considerations raised by the Patent Owner—but I am unaware of any at this time.

VI. GROUND 1: CHERNIN RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS

55. In my opinion, Chernin renders claims 1-3, 5, 8-11, 13, 16-19 obvious.

A. Technical Overview of Chernin

56. Chernin discloses computerized systems and methods for sharing identified cyber-threat information in a standardized and secure format. Chernin at Abstract. The sharing of cyber-threat information assists in preventing malicious actors from replicating successful cyber-attacks by informing potential targets of the methods employed by the malicious actors, and the defensive measures that those targets should implement to prevent those methods from succeeding. *Id.* By distributing cyber-threat information in a standardized format, the systems and methods enable participating entities to automatically analyze and implement defensive measures. *Id.* The systems and methods described by Chernin also permit an entity to control which threat information it shares and which other entities it shares it with. Ex. 1006, [0009].

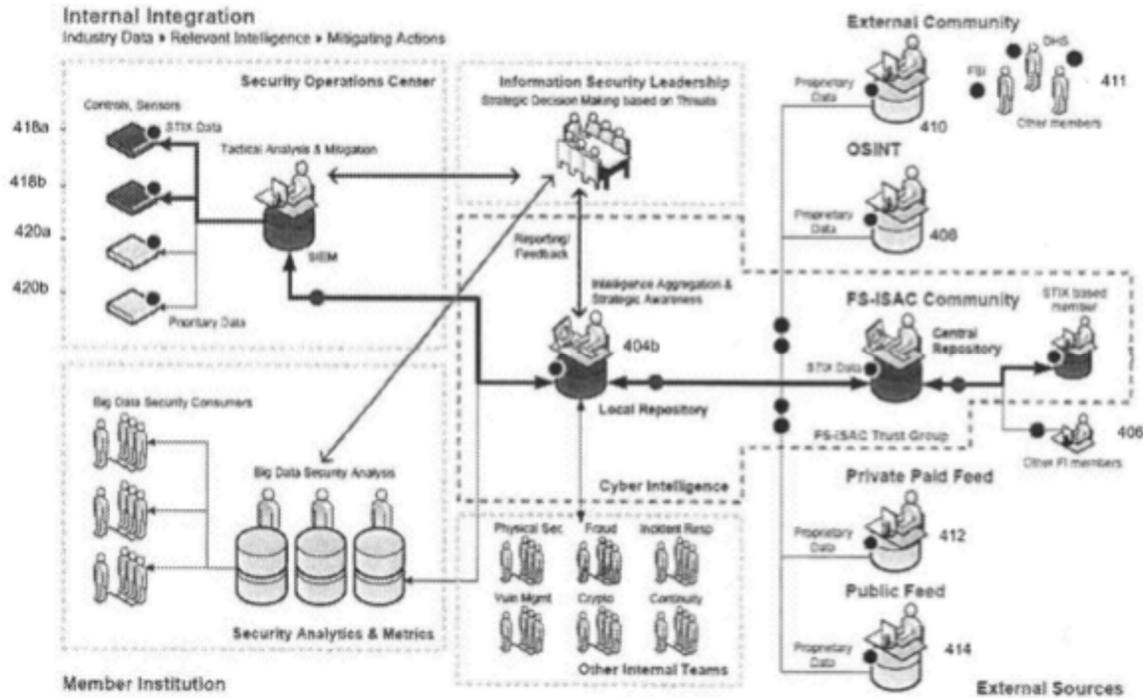
57. As an example, Chernin teaches that a detected malicious IP address can be distributed to other users of the information-sharing system, who can then block that IP address from gaining access to their networks. Ex. 1006, [0006]. A piece of detected malware's filename can be shared with other firms, who can then warn their employees not to download or install the malicious software on their personal devices. *Id.* Therefore, even when a cyber-attacker succeeds, the sharing of information about that successful attack will assist in detecting and preventing any future attempts that cyber-attacker makes. *Id.*

58. Chernin discloses a cross-customer learning model in which multiple participating entities each operate one or more distributed servers. Ex. 1006, [0037]-[0038]. Each of these servers contains a local repository for threat information. *Id.* Chernin discloses two different architectures for sharing threat information among the entities and their respective local repositories. The first is a "hub and spoke" model, in which a central repository is responsible for receiving threat data from the local repositories of the various entities, aggregating and processing it, and sending the processed information back out to entities that have local repositories subscribed to the information. Ex. 1006, [0034]. The second architecture is a "peer to peer" model, by which entities can elect individually which other entities they want to learn from. Ex. 1006, [0059]-[0060]. Chernin describes one particular embodiment in

which there is a directory of “feeds” of items of threat information published by various local repositories and in which an operator can identify which of these feeds should be used for cross-customer learning. Ex. 1006, [0063].

59. Chernin discloses threat information in a standardized format (STIX) transmitted according to a defined specification (TAXII). Ex. 1006, [0068], [0091].

This format allows to encode observable properties, such as information about a file (such as the filename, hash, or size), a registry key value (or the creation or deletion of a key), a service being started or changed, an HTTP request being sent (or received), network traffic on specific IP addresses, and communication on certain ports. Ex. 1006, [0071]. STIX also allows to encode “courses of action (COAs),” which are a representation of one or more specific measures to be taken to address cyberthreats. Ex. 1006, [0082]. A COA can be corrective (acting to address a cyberthreat which has already occurred) or preventative (forward-looking, acting to prevent a future cyberthreat which has not yet occurred). *Id.* Chernin teaches that a COA may include, for example, an intrusion prevention system rule or automatic patch or remediation. *Id.*



(Ex. 1006, Fig. 4)

B. Claims 1, 9, and 17

1[pre]: “A method of learning from other cloud deployments to combat security threats, the method comprising”

9[pre]: “A system for learning from other cloud deployments to combat security threats, the system comprising at least one processor and memory storing computer program instructions that, when executed, cause the system to carry out the steps of”

17[pre]: “A computer program product for learning from other cloud deployments to combat security threats, the computer program product disposed on a non-transitory computer readable medium, the computer program product including computer program instructions that, when executed, carry out the steps of”

60. Chernin discloses or renders obvious limitations 1[pre]/9[pre]/17[pre].

61. Chernin teaches that an entity with a cloud deployment can learn from another entity with a cloud deployment by sharing information about security threats. Abstract (“Computerized systems and methods for sharing identified cyber-threat information in a standardized and secure format. The sharing of cyber-threat information assists in preventing malicious actors from replicating successful cyber-attacks by informing potential targets of the methods employed by the malicious actors, and the defensive measures that those targets should to implement to prevent those methods from succeeding.”); Ex. 1006, Abstract (“The present invention pertains to computer-implemented systems and methods for allowing institutions to securely exchange, distribute, and receive cyber-threat information in a standardized format. More particularly, the present invention is directed to a) a system which allows cyber-threat information to be securely exchanged both between a central repository and distributed local repositories as well as directly between the distributed local repositories, and to b) a method of standardized and securely exchanging cyber-threat information between institutions.”). *Id.*, [0002] Chernin’s Abstract thus recites both a “method” (as in 1[pre]) and a “system” (as in 9[pre]) for sharing threat information. Moreover, Chernin teaches that these systems and methods “can also predict future threats (based on recognized patterns of behavior)

and **preemptively prescribe and apply appropriate preventative and defensive measures** before threats have even been carried out anywhere within the system.”

Ex. 1006, [0033]. This corresponds directly to “learning from other cloud deployments to combat security threats” as recited in the preambles.

62. For example, Chernin recognizes the growing use of computer systems because of “the recent proliferation of mobile devices, **cloud computing**, and the so-called ‘Internet of things.’” *Id.*, [0003]. It recognizes that entities place their data in a repository “in the cloud,” and that this leads to risks such as “the prospect of having [their] information stolen and compromised by a hacker who gains access to that single repository.” *Id.*, [0005]. Chernin discloses that “sharing and distributing cyber-threat information can act as an ‘immune system’ for participating entities—by sharing the source and methods of a cyber-attack detected by one member, the remaining members of the system can be ‘vaccinated’ against that cyber-threat vector, mitigating any future attacks.” *Id.*, [0006]. Chernin further teaches systems and methods for sharing threat information that:

are fully automated, and can not only automatically identify, filter, and respond appropriately to threat information received by other entities, but can also predict future threats (based on recognized patterns of behavior) and preemptively prescribe and apply appropriate preventative and defensive measures before threats have even been carried out anywhere within the system.

Id., [0033].

63. Illustratively, Chernin's FIG. 1 shows a "hub and spoke" system including central repository 102 hosted on a centralized server and member entities 104a-104d, e.g., "individuals, firms, government agencies, and associations," which are hosted on distributed server(s) and connected to the central repository via a network such as the Internet. *Id.*, [0036], [0038].

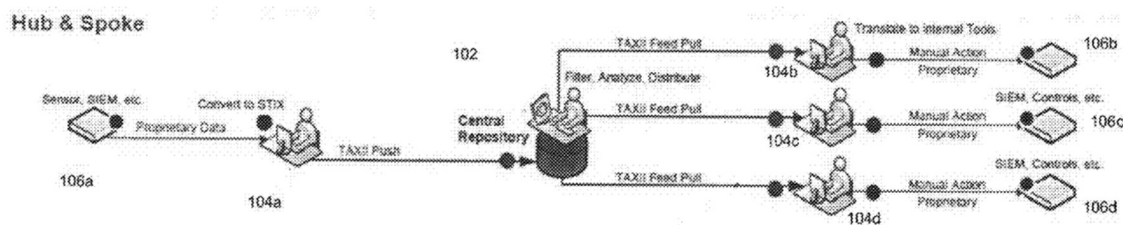


FIG. 1

Id., FIG. 1.

64. In light of Chernin's above disclosures, including paragraphs [0003], [0005], [0006], [0037] and [0038], a POSITA would have immediately understood that [local] "member entities such as individuals, firms, government agencies and associations, each of which are hosted on one or more distributed servers" are "cloud deployments" as recited in the '695 patent claims because each one is hosted on at least one distributed server.

65. Chernin teaches that local entities receive threat information from sensors/controls 106a-106d which "can monitor, identify, and locate known and

unknown cyberattacks and events in real time,” e.g., can detect “intrusions, anomalies, and vulnerabilities.” *Id.*, [0038]. Chernin teaches that when these sensors and controls observe an event or property, they “generate one or more items of threat information describing that event or property in a machine-readable customized language used by that particular local entity.” *Id.*, [0039]. The local entities then “translate the items of threat information from the machine-readable, customized language into a common language used by the system 100 to describe items of threat information” and transmit the items to centralized repository 102. *Id.*, [0040]. The centralized repository receives, stores, and analyzes the items of threat information, and the items of threat information are distributed directly (“hub-and-spoke”) or indirectly (“peer-to-peer”) among the local entities 104a-104d for use in protecting against threats in a manner such as elaborated with reference to other elements of claim 1. *Id.*, [0046]-[0048]. This corresponds precisely to “learning from other cloud deployments to combat security threats” as recited in 1[pre]/9[pre]/17[pre].

66. Chernin discloses performing the relevant operations as part of a “method” as in 1[pre], and a “system” as in 9[pre]. *See, e.g.*, Ex. 1006, Abstract.

67. Chernin discloses that the central server comprises “at least one multi-core processor” hosting a central repository that synchronizes with each of the distributed servers by exchanging threat information. Ex. 1006, [0023]. A POSITA would have

recognized that in order for a processor to perform the listed tasks of “hosting,” “synchronizing,” and “exchanging,” it would be running some kind of program with instructions. Accordingly, a POSITA would have readily recognized that Chernin teaches the “computer program product”.

68. Accordingly, Chernin discloses or renders obvious 1[pre]/9[pre]/17[pre].

1[a]/9[a]/17[a]: “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning;”

69. Chernin discloses or renders obvious “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning,” as recited in 1[a]/9[a]/17[a].

70. In the cloud architectures that Chernin discloses or renders obvious as discussed for 1[pre]/9[pre]/17[pre], [local] [member] “entities such as individuals, firms, government agencies and associations, each of which are hosted on one or more distributed servers” correspond to “cloud deployments”. One of these entities respectively corresponds to a “first cloud deployment” while the others correspond to “one or more additional cloud deployments.”

71. Chernin discloses that the entities can choose who they exchange information with. *E.g.*, Ex. 1006, [0009] (“At the same time, however, because of the sensitive nature of cyber-threat attacks, it is desirable that institutions and individuals have the

ability to control which detected cyber-threats they share, how they share them, and ***which other entities they share them with***, in order to protect against sensitive and sometimes embarrassing disclosures which can damage their privacy, security, and reputation.”).

72. Chernin discloses that threat data can be associated with a quality metric and that the quality metric can be based on a confidence metric. Chernin [0084]-[0085]. The confidence metric can be based on factors including an identity of the entity who was responsible [for creating the threat data]. Chernin continues:

[0086] For example, there may be a high confidence metric and quality metric that the information contained in a construct is accurate and valid if the entity that first observed a threat has a long historical record of accurately characterizing cyberattacks, [...].

[0087] In contrast, if the entity that identified the cyberthreat has just joined the information-sharing network and has no history of correctly identifying cyberthreats [...] the confidence and quality metrics for that STIX language construct will be relatively low.

[0088] ***In certain embodiments of the invention, member entities may choose to respond to constructs with low confidence/quality scores [...] by filtering out the low-quality threat information altogether so that it is ignored.***

73. In other words, Chernin teaches identifying one or more additional cloud deployments to learn from by explicitly providing a mechanism to ignore cloud deployments that provide low-quality threat information. Chernin’s quality- and

confidence-based filtering mechanism is itself a process for “identifying . . . one or more additional cloud deployments to utilize for cross-customer learning,” because the entities whose threat data survives the quality filter are the ones that will be utilized for learning – it is a method of identifying by negative rather than positive selection.

74. Additionally, Chernin discloses that participating entities can form “communities” with an “access control group” for sharing threat information. Ex. 1006, [0057] (“In these embodiments of the invention, the threat information-sharing systems of the present invention may include one or more ‘communities’ of entities and/or repositories, wherein each community may contain one or more access control groups 212.”). An “access control group” “defines a group of entities and/or repositories that are authorized to access certain categories of sensitive or private threat information that could, for example, embarrass an entity or damage that entity’s privacy, reputation, operations, resources, or security.” Ex. 1006, [0054]. Additionally, “each access control group 212 [can] designat[e] a specific group of entities as being able to access information of varying privacy levels from the other entities in that group 212.”

75. The communities may be created and/or operated by an administrator and may be a “public community (a community visible to non-member entities and/or

repositories and that does not require permission from an administrator to join), a private community (a community visible to non-member entities and/or repositories and that requires permission from an administrator to join), or a secret community (a community invisible to non-member entities and/or repositories that requires permission from an administrator to join).”). *Id.*

76. These groups allow an administrator to customize information sharing between entities. Ex. 1006, [0055]. For example, in Figure 2, access control group 212 includes “local repositories 204a, 204b, and 204c and central repository 202, while excluding local entity 210.” *Id.* In this example, if a piece of threat information would be damaging if revealed to the general public, it would be shared with repositories 202, 204a, and 204b, but not local entity 210. *Id.* As another example of identifying entities for cross-learning, “under the TLP standard, information coded as ‘RED’ [may] not to be shared with any parties outside of the specific exchange, meeting, or conversation in which it was initially disclosed.” Ex. 1006, [0058].

77. Additionally, Chernin discloses that member entities can subscribe to each other’s threat feeds. Ex. 1006, [0063] (“In some of the above-mentioned peer-to-peer embodiments of the invention, the local repositories 304a, 304b, and 304c may each publish their own ‘feed of items of threat information to be accessed by other local repositories 304a, 304b, and 304c. In some embodiments, all or a subset of

these feeds may be registered in a directory (not pictured) that allows an operator of a local repository 304a, 304b, or 304c to access a list of other local repository feeds to connect or subscribe to.”). An operator may identify different feeds to subscribe to (“operator of a local repository 304a, 304b, or 304c may browse through a list of all feeds registered in the directory; search for feeds in the directory that satisfy a set of search criteria: and/or view one or more feed(s) suggested by the directory itself based on the similarity of the operator’s repository and the repositories that provide those suggested feeds.”) and may “push” or “pull” information from the other entities (“The operator can select one or more feeds from the registered directory to “subscribe” to, causing those one or more subscribed-to feeds to automatically “push” all or a subset of their published items of threat information to the operators repository, or to cause the operator’s repository to automatically “pull” all or a subset of the published items of threat information from the subscribed-to feeds.”). Ex. 1006, [0063]. While Chernin does not explicitly disclose the centralized server being part of this subscription process, a POSITA would have understood that “hub-and-spoke” is a simpler networking architecture than a general peer-to-peer architecture, and that Chernin’s discloses both networking architectures. In the hub-and-spoke architecture, local entities are connected to and share information with the central repository. A POSITA would have found it obvious that in this simpler and more

restrictive networking architecture, the feed directory would be part of the centralized server since all of the local entities connect to it directly, while they do not connect to each other directly. This setup would allow the feeds to be accessed and managed from a central location.

78. A POSITA would also have readily understood or found obvious that Chernin's cross-customer learning approach would not always apply to the entirety of a first cloud deployment but sometimes only to "a portion of a first cloud deployment". This is because many categories of threat information are specific to a particular hardware platform or operating system. For example, if a first cloud deployment contained one server running Ubuntu Linux and another server running Windows Server, then certain categories of cross-customer learning would apply only a portion of the first cloud deployment in which specific features of that portion of the cloud deployment matched the identical features in the additional cloud deployments being learned from. There is usually very little correlation between specific vulnerabilities in different versions of different operating systems.

79. Accordingly, Chernin discloses or renders obvious 1[a]/9[a]/17[a].

1[b]/9[b]/17[b]: "receiving information describing a security threat to one or more of the additional cloud deployments;"

80. Chernin discloses or renders obvious “receiving information describing a security threat to one or more of the additional cloud deployments,” as in 1[b]/9[b]/17[b].

81. As discussed above with reference to 1[pre]/9[pre]/17[pre], Chernin discloses that entities (cloud deployments) receive threat information from sensors/controls 106a-106d which “can monitor, identify, and locate known and unknown cyberattacks and events in real time,” e.g., can detect “intrusions, anomalies, and vulnerabilities.” Ex. 1006, [0038]. Chernin teaches that when these sensors and controls observe an event or property, they “generate one or more items of threat information describing that event or property in a machine-readable customized language used by that particular local entity.” *Id.*, [0039]. The entities then “translate the items of threat information from the machine-readable, customized language into a common language used by the system 100 to describe items of threat information” and transmit the items to centralized repository 102. *Id.*, [0040]. The centralized repository receives, stores, and analyzes the items of threat information, and the items of threat information are distributed directly or indirectly among the local entities 104a-104d for use in protecting against threats. *Id.*, [0046]-[0048]. This corresponds precisely to “receiving information describing a security threat to one or more of the additional cloud deployments” as recited in 1[b]/9[b]/17[b]. For

example, Chernin discusses “SIEM sensors ... observ[ing] an event or property.” *Id.*, [0039]. The local entity takes that observable event and generates an item of threat information in its local language. A POSITA would have understood that this is the event mentioned in [0013] and [0039] and that it is translated into, for example, the “Observable” event in STIX mentioned in [0071].

82. Accordingly, Chernin discloses or renders obvious 1[b]/9[b]/17[b].

1[c]/9[c]/17[c]: “receiving information describing configuration settings used to combat the security threat; and”

83. Chernin discloses or renders obvious “receiving information describing configuration settings used to combat the security threat,” as recited in 1[c]/9[c]/17[c].

84. The '695 patent provides the following examples of what such information may include:

The configuration settings used to combat the security threat may be embodied, for example, as configuration settings that caused the additional cloud deployments 912 to blacklist certain IP addresses or network domains, as configuration settings that caused the additional cloud deployments 912 to close vulnerabilities that allowed an attack to succeed, or some other configuration that would be useful in detecting/preventing/mitigating a security threat.

Ex. 1001, 81:43-51.

85. Chernin discloses that entities store and distribute threat information to one another and/or to the centralized repository using a “common language,” such as “Structured Threat Information eXpression (STIX) language.” Ex. 1006, [0034] (“In this type of system 100, one or more central repositories 102 and a plurality of users 104a, 104b, 104C, and 104d communicate items of threat information in a common language and distribute those items of threat information in accordance with a common specification.”), [0039] (“After generating the one or more items of information, the local entities 104a, 104b, 104C, and 104d translate the items of threat information from the machine readable, customized language into a common language used by the system 100 to describe items of threat information. Once translated, the items of threat information are then transmitted from each of the local entities 104a, 104b, 104c, and 104d to centralized repository 102 in accordance with a common specification for the transmission of information.”), [0046] (“In addition to analyzing the items of threat information that it has received and stored, the central repository 102 is also capable of distributing items of threat information to local entities 104a, 104b, 104c, and 104d.”), [0068] (“In certain embodiments of the invention, the common language used by the plurality of repositories for storing and distributing threat information is the Structured Threat Information eXpression

(STIX) language.”). The STIX language includes “Observable,” “Incident” and “CourseOfAction (COA)” constructs, among others:

The “base” construct of the STIX language is the “Observable.” Observables are stateful properties or measurable events pertinent to the operation of computers and networks. Examples of Observables include (but are not limited to): information about a file (such as the filename, hash, or size), a registry key value (or the creation or deletion of a key), a service being started or changed, an HTTP request being sent (or received), an email being received (or sent), ***network traffic on specific IP addresses***, or communication on certain ports. In certain embodiments, STIX utilizes the Cyber Observable eXpression schema (CybOX) to represent Observables.

Incidents are discrete instances in which an Indicator affects an organization, **along with information discovered or decided during the investigation into how to respond to the Incident**. Incidents include (for example),...the nature of what (if any) was compromised in the Incident, **the responsive Course of Action [COA] (another type of construct) taken**, the confidence in the characterization in the incident, guidance on handling the incident, the source of the Incident information, and a **log of actions taken during and in response to the Incident**.

A COA can be corrective (acting to address a cyberthreat which has already occurred) or preventative (forward-looking, acting to prevent a future cyberthreat which has not yet occurred) either to address one or more ExploitTargets or to counter or mitigate the impacts of Incidents. A COA includes (but is not limited to) its relevant stage in cyberthreat management (e.g., remedy of an ExploitTarget or response to an Incident), the type of the COA, a description of the COA, the objective of the COA, the structured representation of

the COA (such as an IPS rule or automated patch or remediation), the likely impact of the COA, the likely cost of the COA, the estimated efficacy of the COA, observable parameters of the COA, and handling guidance.

Id., [0071], [0074], [0082].

86. Accordingly, the STIX language includes a log of actions taken during and in response to an Incident, and a COA that was taken to address a cyberthreat that occurred. *Id.* A POSITA would have readily recognized that the COA’s structured representation such as an “IPS rule or automated patch or remediation” describes configuration settings used to combat the security threat. An IPS (Intrusion Prevention System) rule is a configuration directive that instructs the intrusion prevention system to inspect network traffic and block or drop packets matching specified malicious patterns—it is a configuration setting applied to a network security appliance.

87. Likewise, an “automated patch or remediation” modifies the software or settings of a system to close a known vulnerability—another form of configuration change. Chernin separately teaches a concrete example: “a detected malicious IP address can be distributed to other users of the information-sharing system, who can then **block that IP address from gaining access to their networks.**” Ex. 1006, [0006]. Blocking a malicious IP address requires adding that address to a firewall blacklist or access-control list, which is itself a configuration setting. As such, when

the local entities (cloud deployments) send threat information to the central repository (*see* 1[b]/9[b]/17[b]) using STIX, the central repository and/or another local entity “receiv[es] information describing configuration settings used to combat the security threat,” just as recited in 1[c]/9[c]/17[c].

88. Additionally, Chernin discloses that the COA can include more than one configuration setting. Ex. 1006, [0082] (“Another construct in the STIX language is a CourseCfAction (COA), which is a representation of *one or more specific measures* to be taken to address cyberthreats.”).

89. Accordingly, Chernin discloses or renders obvious 1[c]/9[c]/17[c].

1[d]/9[d]/17[d]: “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.”

90. Chernin discloses or renders obvious “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud discloses,” as recited in 1[d]/9[d]/17[d].

91. Chernin teaches that the local entities send threat information to the central repository or to one another using STIX (*see* 1[c]/9[c]/17[c]). The central repository or other local entity (each of which is associated with a local repository) thus “receiv[es] information describing configuration settings used to combat the security threat,” particularly the COA’s representation such as an “IPS rule or automated

patch or remediation.” Ex. 1006, [0082]. Indeed, the entire purpose of the COA in the STIX data is to represent “one or more specific measures to be taken to address cyberthreats.” *Id.*

92. Chernin further teaches that the central repository identifies, based on the threat information it receives (in the hub-and-spoke architecture), one or more configurations to adopt for the local repositories: “the analysis performed by central repository 102 may allow the central repository 102 to **determine...a particular action that could be taken to prevent, mitigate, or remediate the effects of at least one of the multiple observed events or properties, the entity responsible for at least one of the multiple observed events or properties.**” Ex. 1006, [0044]. The central repository then distributes items of threat information to the local entities via “pull” or “push,” and in some embodiments “is capable of determining unique subsets of the items of threat information to be distributed to each of the local entities.” *Id.*, [0047]-[0048].

93. Chernin teaches a wide variety of actions that can be taken to prevent, mitigate, or remediate the effects of cyberattacks that have already occurred, e.g.,

A system for sharing and distributing cyber-threat information can act as an “immune system” for participating entities—by sharing the source and methods of a cyber-attack detected by one member, the remaining members of the system can be “vaccinated” against that cyber-threat vector, mitigating any future attacks. For

example, a detected malicious IP address can be distributed to other users of the information-sharing system, who can then block that IP address from gaining access to their networks. A piece of detected malware's filename can be shared with other firms, who can then warn their employees not to download or install the malicious software on their personal devices. Therefore, even when a cyber-attacker succeeds, the sharing of information about that successful attack will assist in detecting and preventing any future attempts that cyber-attacker makes.

Ex. 1006, [0006].

94. Based on Chernin's teachings, a POSITA readily would have recognized that based on the STIX data that the central repository receives from a local entity that has been attacked, the central repository identifies configurations—such as blocking a malicious IP address or setting a warning to not download or install malicious software—that another local entity should adopt to reduce the threat. Indeed, that is the reason for sharing the STIX data with the central repository. In particular, the COA construct's structured representation (e.g., an IPS rule or automated patch) identifies configurations to adopt—the very purpose of the COA is to represent “one or more specific measures to be taken to address cyberthreats.” Ex. 1006, [0082]. Chernin further teaches that the central repository's analysis results, including “a particular action that could be taken to prevent, mitigate, or remediate the effects of at least one of the multiple observed events or properties,” are added back to the threat items that are shared. Ex. 1006, [0016] (the computerized methods “further

comprising adding the at least one piece of information to at least one of the plurality of items”); *see also* Ex. 1006, Claim 16. Thus, when the central repository identifies a remediation action (e.g., a firewall rule or patch), it is able to annotate an existing threat item with that action, and such annotated items flow to the other local entities via the existing STIX/TAXII distribution infrastructure. Moreover, a POSITA would have recognized that based on the threat information that a local entity receives (whether from the central repository or from a local repository associated with another local entity), that local entity can identify and then implement configurations to reduce the threat. Likewise, this is the reason for sharing the threat information with the local entity via, for example, the COA construct.

95. As such, Chernin discloses or renders obvious “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment,” as recited in 1[d]/9[d]/17[d].

1. Claims 2/10/18

2[pre]: “The method of claim 1” / 10[pre]: “The system of claim 9” / 18[pre]: “The computer program product of claim 17”

96. Claim 2 depends from claim 1; claim 10 depends from claim 9; and claim 18 depends from claim 17. Chernin discloses or renders obvious claims 1/9/17.

2[a]/10[a]/18[a]: “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.”

97. Chernin discloses or renders obvious “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses,” as recited in 2[a]/10[a]/18[a].

98. As discussed for 1[d]/9[d]/17[d], Chernin teaches that “a detected malicious IP address can be distributed to other users of the information-sharing system, who can then block that IP address from gaining access to their networks” Ex. 1006, [0006] and that the “‘base’ construct of the STIX language” used to communicate threats can include “network traffic on specific IP addresses” *Id.*, [0071].

99. From such teachings, a POSITA would have found it obvious that the STIX data shared with the central repository and/or the threat information shared with the local entities (which correspond to cloud deployments as discussed for 1[pre]/9[pre]/17[pre]) identify a previously-identified malicious IP address to add to a blacklist, and that such identification corresponds to “configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses,” as recited in 2[a]/10[a]/18[a].

2. Claims 3/11/19

3[pre]: “The method of claim 1” / 11[pre]: “The system of claim 9” / 19[pre]: “The computer program product of claim 17”

100. Claim 3 depends from claim 1; claim 11 depends from claim 9; and claim 19 depends from claim 17. Chernin discloses or renders obvious claims 1/9/17.

3[a]/11[a]/19[a]: “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains.”

101. Chernin discloses or renders obvious “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains,” as recited in 3[a]/11[a]/19[a].

102. Chernin discloses or renders obvious this claim for reasons similar to claim 2. The base “Observable” construct includes “an HTTP request being sent (or received).” Ex. 1006, [0071]. A POSITA would have found it obvious, for example, that HTTP requests contain URLs, which can contain malicious domain names (among other information) that the attacked local entity (which corresponds to a cloud deployments as discussed for 1[pre]/9[pre]/17[pre]) has identified. Moreover, the POSITA would have understood that Chernin’s threat information and STIX data are designed so that the local entities can “prevent, mitigate, or remediate the effects

of the observable event or property, *e.g.*, block the previously-identified malicious network domain based on the domain name (among other things).

103. Accordingly, Chernin discloses or renders obvious 3[a]/11[a]/19[a].

3. Claims 5/13

**5[pre]: “The method of claim 1 further comprising” /
13[pre]: “The system of claim 9, wherein the steps
further comprise”**

104. Claim 5 depends from claim 1; and claim 13 depends from claim 9. Chernin discloses or renders obvious claims 1/9.

**5[a]/13[a]: “receiving information describing one or
more actions associated with the additional cloud
deployments; and”**

105. Chernin discloses or renders obvious “receiving information describing one or more actions associated with the additional cloud deployments,” as recited in 5[a]/13[a].

106. The ’695 patent describes “action:”

Likewise, a message, request, or other form of data communications that are exchanged between the additional cloud deployments 912 and actors that are external to the additional cloud deployments 912 may be an example of an action that is associated with the additional cloud deployments 912.

Ex. 1001, 79:41-46.

107. A POSITA would have understood that any sudden communication activity between any local entity and an external third party might be indicative of local malware communicating outward with a command-and-control node or indicative of an outside party communicating inward attempting to exploit a software vulnerability. The POSITA would have also understood that if such an observed action was later confirmed as a security attack, a possible approach to prevent the same attack in other local entities would be to configure the firewalls of those other local entities to disallow any future communication with the suspicious external third party actor. Additionally, Chernin’s “Incident” construct expressly includes “a log of actions taken during and in response to the Incident.” Ex. 1006, [0074]. This “log of actions” is a direct analog to “information describing one or more actions associated with the additional cloud deployments” as recited in 5[a]/13[a], and it is part of the STIX data that is shared among the local entities and the central repository.

108. Chernin includes information about such communication in the STIX data that local entities send to the central repository or to each other:

The “base” construct of the STIX language is the “Observable.” Observables are stateful properties or measurable events pertinent to the operation of computers and networks. Examples of Observables include (but are not limited to): information about a file (such as the filename, hash, or size), a registry key value (or the creation or deletion of a key), a service being started or changed, an HTTP request being sent (or received), an

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

email being received (or sent), network traffic on specific IP addresses, or communication on certain ports.

Ex. 1006, [0071].

109. The '695 patent also describes another example of an “action”:

In other embodiments, the one or more actions associated with the additional cloud deployments 912 may include one or more user interactions with the additional cloud deployments 912. Such user interactions can include a description of which resources are accessed and/or utilized by particular users, particular business units (e.g., the finance department of a business, the engineering department of a business), particular personas (e.g., a system administrator, a software developer, a human resources manager), and so on.

Ex. 1001, 79:51-59.

110. According to Chernin,

... items of threat information may be automatically or manually generated at the local entities 104 a, 104 b, 104 c, and 104 d, and further details and metadata can be added to a particular item of information generated by the local entities 104 a, 104 b, 104 c, and 104 d.

Ex. 1006, [0039].

111. A POSITA would understand the “manual” addition of threat information to constitute a user interaction with a local entity (i.e., one of Chernin’s additional cloud deployments). In Chernin’s system, such manually added items of threat information are then automatically distributed, i.e., they are subsequently received.

112. Accordingly, Chernin discloses or renders obvious 5[a]/13[a].

5[b]/13[b]: “wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions.”

113. Chernin discloses or renders obvious “wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions,” as recited in 5[b]/13[b].

114. As discussed above in for claims 1/9/17, Chernin discloses or renders obvious “identifying...one or more configurations to adopt for the first cloud deployment.”

And, as explained above with reference to 5[a]/13[a], Chernin discloses or renders obvious that the threat information describes one or more actions associated with the additional cloud deployments, e.g., in the form of the “Observable” construct in STIX language. Chernin discloses or renders obvious that configuration(s) to adopt for a given local entity (first cloud deployment) are based on the action(s) of another local entity (additional cloud deployment). Indeed, STIX includes several nested constructions based on the Observables, such as “Indicator” which “conveys information about specific patterns of Observables, combined with contextual information, which is intended to represent behaviors of interest within the cybersecurity context; ”Incident“ which is a discrete instance ”in which an Indicator affects an organization, along with information discovered or decided during the investigation into how to respond to the incident,“ including ”related Observables;

and “CourseOfAction (COA)” which represents “one or more specific measures to be taken to address cyberthreats” and is based indirectly on Observable and Indicator constructs, among other interrelated constructs.” Ex. 1006, [0073]-[0083]; *see* 1[c]/9[c]/17[c].

115. A POSITA readily would have recognized that based on the STIX data—including such information about actions—that Chernin’s central repository receives from a local entity that has been attacked, the central repository is able to use the Incident, Observables, COA, and other constructs to identify configurations that another local entity should adopt to reduce the threat. Indeed, that is a reason for sharing the STIX data with the central repository. Moreover, a POSITA would have recognized that based on the threat information—including information about actions, per 5[a]/13[a]—that a local entity receives (whether from the central repository or directly from a repository associated with another local entity), that local entity can identify and then implement configurations to reduce the threat. Likewise, this is a reason for sharing the threat information with the local entity.

116. Accordingly, Chernin discloses or renders obvious 5[b]/13[b].

5. Claims 8/16

8[pre]: “The method of claim 1 further comprising” /
16[pre]: “The system of claim 9 wherein the steps
further comprise”

117. Claim 8 depends from claim 1; and claim 16 depends from claim 9. Chernin teaches claims 1/9.

8[a]/16[a]: “receiving information describing one or more deployment processes associated with the additional cloud deployments.”

118. Chernin discloses or renders obvious “receiving information describing one or more deployment processes associated with the additional cloud deployments,” as recited in 8[a]/16[a].

119. Chernin teaches the very thing that Fortinet accuses of infringing claims 8/16.

120. More specifically, in its ICs (Ex. 1016), Fortinet alleges:

Netskope Cloud Exchange may receive information from third party cloud deployments that can include ***information concerning the name of the file associated with an identified threat***, which is insubstantially different from information concerning the deployment processes employed in the cloud deployment as the file name (as incorporated in the “comment” field) can establish the component (e.g., program) within the cloud deployment associated with the identified threat.

Infringement Contentions at 43. The “Observable” construct in STIX—which conveys structured cyber-threat information—includes “information about a file (such as the filename, hash, or size).” Ex. 1006, [0070]-[0071].

121. Accordingly, the filename is among that which the centralized repository receives from local entities that have been attacked, and that local entities receive from each other when they share threat information directly.

122. Accordingly, Chernin discloses or renders obvious 8[a]/16[a].

VII. GROUND 2: THE COMBINATION OF CHERNIN AND JEUK RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS

123. In my opinion, the combination of Chernin and Jeuk renders claims 1-3, 5, 8-11, 13, 16-19 obvious.

A. Technical Overview of Jeuk

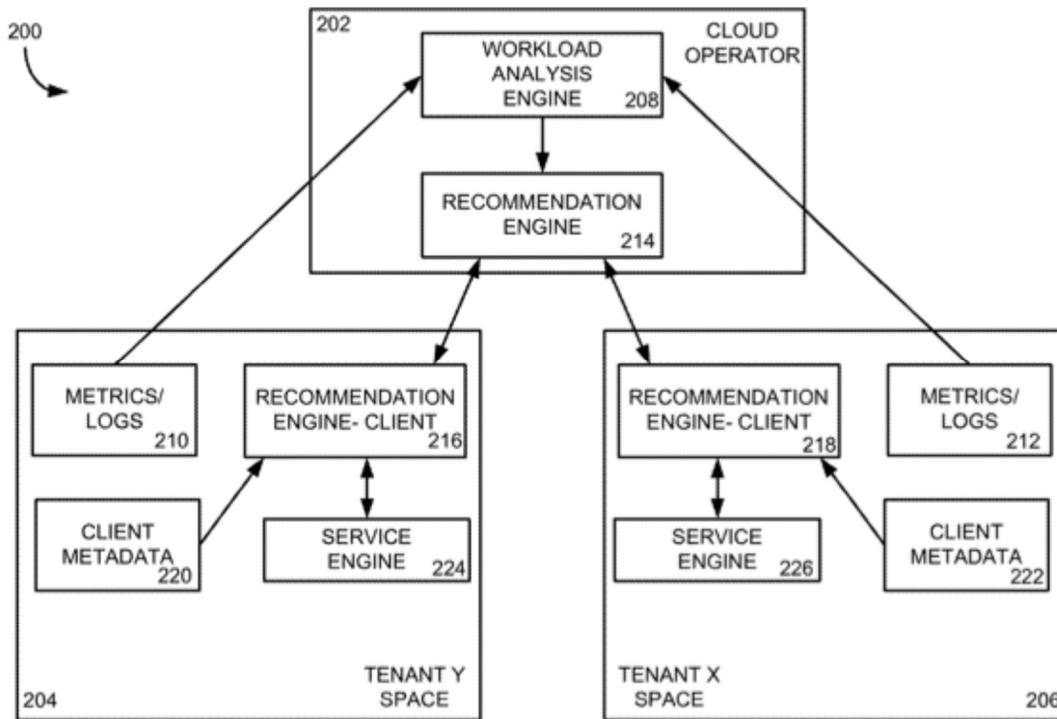
124. Jeuk discloses systems and methods for receiving and correlating workload performance information from multiple customers in separate “tenant spaces,” and, by using cross-customer learning, providing deployment improvement suggestions to a cloud operator or customer based at least on the correlated workload performance information. Ex. 1007, Abstract [0017].

125. In particular, Jeuk discloses a “workload analysis engine” that receives a first set of metrics and log information from a first tenant space and create a first performance pattern from this first data set. Ex. 1007, [0014], [0027]. The workload analysis engine also receives a second set of metrics and log information from a second tenant space and create a second performance pattern from the second data set. *Id.* The workload engine may then utilize the received performance information

and one or more machine learning approaches to detect performance patterns of an application across the different tenant deployments. Ex. 1007, [0017]. In other words, the engine may identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants. *Id.*

126. A recommendation engine in communication with the workload analysis engine then correlates the received metrics and log information and uses it to generate deployment recommendations. Ex. 1007, [0018], [0037]. The recommendation engine may analyze the performance characteristics across the multiple tenant applications and determine a “best fit” or optimized deployment of the application. Ex. 1007 [0018].

127. Once an optimized deployment has been generated, the recommendation engine may provide the optimized deployment to a cloud environment administrator and/or one or more of the tenants. Ex. 1007, [0019]. The optimized deployment may represent a suggested improvement to the deployed application and may be sent to a client recommendation engine within a tenant space. *Id.* The client recommendation engine may then determine if the recommended optimized deployment for the application should or should not be executed within the tenant space. *Id.*



(Ex. 1007, Fig. 2)

B. Jeuk's Role in Prosecution History

128. During prosecution of the '695 patent's parent application (Serial No. 17/671,199, titled "Learning from Similar Cloud Deployments"), the examiner relied heavily on Jeuk to push back on the novelty of the claims.

129. The examiner rejected claims 1-2, 9-10, and 17-18 under 35 U.S.C. § 102(a)(1) as anticipated by Jeuk alone, and rejected claims 3-6, 8, 11-14, 16, and 19-20 under 35 U.S.C. § 103 as obvious over Jeuk in combination with secondary references. Jeuk served as the primary reference in every rejection. Ex. 1005, 318-332.

130. The examiner found that Jeuk anticipated several core limitations of the parent application's independent claims. Specifically, the examiner found that Jeuk disclosed the following:

Limitation (Parent App Claim 1)	Examiner's Basis in Jeuk
Preamble: "A method of learning from similar cloud deployments"	Jeuk generally
1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning"	Jeuk Abstract – workload engine detects performance patterns across different tenant deployments
1[b]: "receiving information describing configurations associated with the additional cloud deployments"	Jeuk [0036] – workload analysis engine correlates identified performance pattern with a particular configuration of cloud resources
1[c]: "identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment"	Jeuk [0018] – recommendation engine determines a "best fit" or optimized deployment
Claim 8: "receiving information describing one or more deployment processes associated with the additional cloud deployments"	Jeuk Abstract – applications provide performance logs and metrics to inter-tenant workload engine

131. The examiner acknowledged that Jeuk did not teach limitations related to security threats, configuration settings used to combat security threats, user permissions, over-permissioned users, or reduced privilege levels. For those limitations (then-claims 3-6), the examiner relied on a secondary reference.

132. The applicant overcame the rejections by amending independent claims 1, 9, and 17 to incorporate subject matter from canceled dependent claims 5 and 6 (and their system/computer program product counterparts, claims 13 and 14). *Id.*, 297-308. The amended claims added limitations directed to receiving information describing permissions for one or more users of the additional cloud deployments, determining that one or more users of the first cloud deployment are over-permissioned, and identifying a reduced privilege level to give to the one or more users. In light of the amendments, the examiner allowed the claims of the parent application. *Id.*, 264-266.

133. The limitations that the examiner found Jeuk disclosed during the parent prosecution closely correspond to the limitations recited in the Challenged Claims of the '695 patent. The following table juxtaposes these limitations:

Limitation Found in Jeuk (Parent App)	Corresponding '695 Patent Claim Limitation
Preamble: "A method of learning from similar cloud deployments"	1[pre]: "A method of learning from other cloud deployments to combat security threats"
1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning"	1[a]: "identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning" (identical)

1[b]: “receiving information describing configurations associated with the additional cloud deployments”	1[b]: “receiving information describing a security threat to one or more of the additional cloud deployments” (narrowed to security context)
1[c]: “identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment”	1[d]: “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment” (security-specific)
Claim 8: “receiving information describing one or more deployment processes associated with the additional cloud deployments”	Claim 8: “receiving information describing one or more deployment processes associated with the additional cloud deployments” (identical)

Ex. 1005, pg. 320-322, 328 and Ex. 1001, 93:19-65.

134. As this comparison demonstrates, the core structural framework of the ’695 patent’s Challenged Claims — identifying additional cloud deployments for cross-customer learning, receiving information from those deployments, and identifying configurations to adopt — is the same framework that the examiner found disclosed by Jeuk during prosecution of the parent.

135. The ’695 patent’s Challenged Claims essentially add a security-threat overlay to Jeuk’s cross-tenant learning architecture: the preamble adds “to combat security threats,” and the body limitations specify receiving a “security threat” and

“configuration settings used to combat the security threat.” Notably, the subject matter that actually distinguished the parent claims from Jeuk — user permissions, over-permissioned users, and reduced privilege levels — does not appear in any of the ’695 patent’s Challenged Claims.

C. Motivation to Combine Jeuk and Chernin.

136. A POSITA at the time of the alleged invention would have been motivated to combine the disclosures of Jeuk and Chernin, with a reasonable expectation of success.

137. While it is my opinion that Chernin discloses “cloud deployments,” to the extent it does not, a POSITA implementing Chernin’s threat-information-sharing scheme would have immediately recognized the benefits of deploying the scheme in cloud deployments, as disclosed in Jeuk. Before the earliest priority date of the ’695 patent (November 2017), cloud deployments, which typically include public clouds, private clouds, and hybrid clouds, among others, were playing an increasingly prevalent role for computing resources, infrastructure, security, etc., particularly for enterprises, as I discussed in the background section. A POSITA would have known that cloud deployments were an important part of current and future computing and security and would have been motivated to deploy Chernin’s scheme in the cloud. Indeed, Chernin explicitly recognizes the added risks that comes with placing

organizational information “in the cloud”. (“An institution that has placed its data in a repository ‘in the cloud’ now faces the prospect of having its information stolen and compromised by a hacker who gains access to that single repository.” Chernin at [0005].)

138. Additionally, a POSITA implementing Chernin’s threat-information-sharing scheme would have immediately recognized that Jeuk’s pattern correlation and recommendation approach could be advantageously modified to optimize defenses for Chernin’s cloud deployments. A POSITA would have been highly motivated to combine Jeuk’s correlation and recommendation techniques into Chernin and adapt them for the specific objectives of cyber defense. Combining Jeuk into Chernin in this manner would have had a reasonable expectation of success, since Chernin and Jeuk share a similar architecture in which a central authority collects information from cloud deployments, processes this information, and then communicates back to at least a subset of the cloud deployments in reaction to the processing of the originally collected information.

139. A POSITA would understand that Chernin gathers, translates, and distributes threat information in this manner, while Jeuk gathers performance data, calculates deployment improvements from this data, and then transmits recommendations based on calculated improvements. A POSITA simultaneously in possession of

Chernin and Jeuk would immediately see the usefulness of and be highly motivated to apply Jeuk's concept of "recommending calculated improvements" to Chernin's subject matter of threat information. Jeuk itself recognizes that its invention can be utilized to provide recommendations of "other applications or workloads." Ex. 1007, [0062].

140. Making such a modification would provide additional benefits to Chernin while changing nothing substantive about how Chernin's threat-sharing functions operate – making this upgrade to Chernin would provide an additional capability to Chernin without diminishing its existing capabilities. The modification would add Jeuk's correlation and recommendation features to Chernin's central repository, which would have been straightforward since this functionality needs to be added in a single location. From a technical standpoint, integrating Jeuk's workload analysis engine (which "analyzes the performance and associated patterns of applications run or executed by different tenants," Ex. 1007, [0026]) into Chernin's central repository is a straightforward software engineering task of adding an additional data processing module at an existing data processing location without changing the overall data flow architecture—the central repository already ingests, stores, and processes STIX-formatted threat data, and Jeuk's engine would operate on that same data to identify cross-tenant patterns and generate recommendations. Additionally, one would need

a mechanism for distributing the resulting recommendations back to at least a subset of client cloud deployments. Notably, such a distribution function from a central repository to individual cloud deployments already exists in Chernin for the threat information via the TAXII protocol. Under these circumstances, adding further data to be sent via the existing distribution infrastructure would also have been very straightforward—no new communication protocol or network architecture would be required.

141. Indeed, Chernin already discloses or suggests a similar “correlation and recommendation” approach. Specifically, Chernin teaches that the central repository can analyze received threat information, including “*comparing the received items of threat information to previously stored items of threat information*, adding additional metadata to the received items of threat information, *determining patterns in the received and stored items of threat information*, forming and dissolving relationships between items of threat information, and *making future predictions based on the received and stored items of threat information present at central repository 102*.” Ex. 1006, [0043]. From this analysis, the central repository can identify vulnerabilities and remediation actions:

For example, the analysis performed by central repository 102 may allow the central repository 102 to determine a representation of the behavior or modus operandi a representation of the behavior or modus operandi of an entity responsible for at least one of the multiple

observed events or properties, a vulnerability, weakness, or configuration issue of a victim of at least one of the multiple observed events or properties, ***a particular action that could be taken to prevent, mitigate, or remediate the effects of at least one of the multiple observed events or properties***, the entity responsible for at least one of the multiple observed events or properties; and/or a prediction of at least one future event or property connected to the already-observed items of threat information.

Ex. 1006, [0044].

142. Chernin teaches that the results of such analyses can be encoded in the STIX format (Ex. 1006, [0013]) and that such analysis results can be added to the information items that are being shared. (Ex. 1006, [0016], *Id.*, Claim 16).

143. Thus, it would have been straightforward to a POSITA to combine Chernin and Jeuk, and the resulting combination would provide an additional capability to Chernin without diminishing its existing capabilities.

D. Claims 1, 9, and 17

1[pre]: “A method of learning from other cloud deployments to combat security threats, the method comprising”

9[pre]: “A system for learning from other cloud deployments to combat security threats, the system comprising at least one processor and memory storing computer program instructions that, when executed, cause the system to carry out the steps of”

17[pre]: “A computer program product for learning from other cloud deployments to combat security threats, the computer program product disposed on a non-transitory computer readable medium, the

**computer program product including computer
program instructions that, when executed, carry out
the steps of”**

144. For the reasons I explained in Ground I, Chernin along discloses or renders obvious limitations 1[pre]/9[pre]/17[pre].

145. But to the extent Chernin does not specifically disclose sharing information between “cloud deployments,” a POSITA would have found it obvious to deploy Chernin in such deployments in view of Jeuk.

146. As I discussed above, the examiner found that Jeuk disclosed the concept of learning from cloud deployments during examination of the ’695 patent’s parent.

147. Jeuk has a similar architecture to Chernin’s “hub-and-spoke” model and similar goal in which tenants on a cloud (“cloud deployments”) learn from one another to improve computing performance. Illustratively, Jeuk discloses in FIG. 1, reproduced below, that:

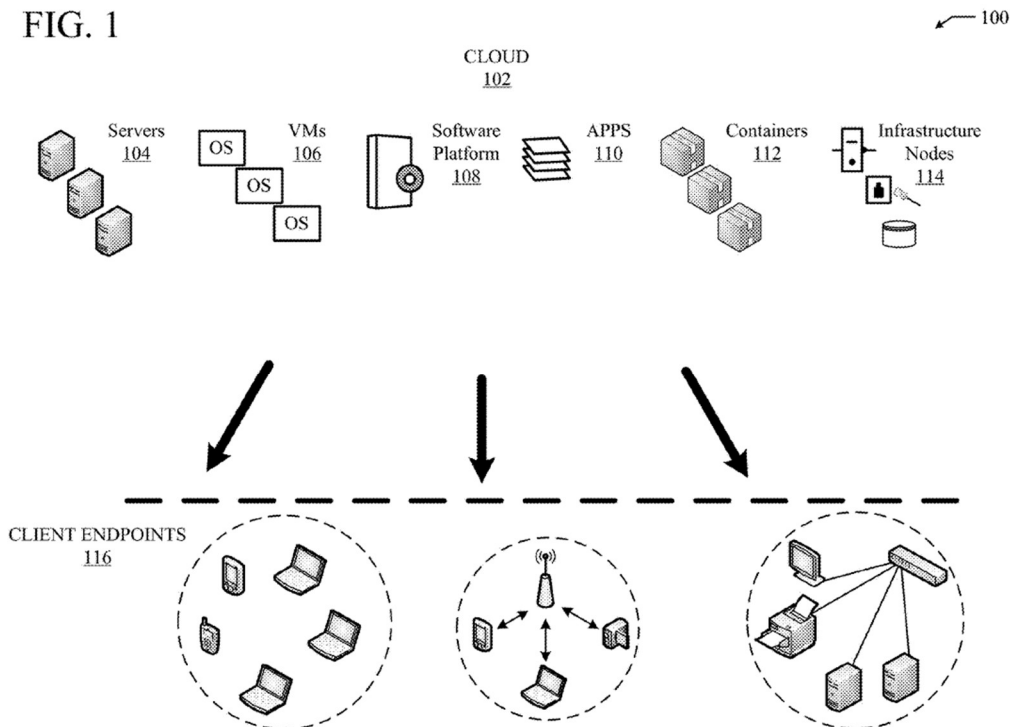
The architecture can include a cloud computing environment 102. The cloud 102 may include one or more private clouds, public clouds, and/or hybrid clouds. Moreover, the cloud 102 may include any number and type of cloud elements 104-114, such as servers 104, virtual machines (VMs) 106, one or more software platforms 108, applications or services 110, software containers 112, and infrastructure nodes 114....

The cloud 102 may provide various cloud computing services via the cloud elements 104-114 to one or more clients 116 of the cloud environment. For example, the

cloud environment 102 may provide software as a service (SaaS)..., infrastructure as a service (IaaS)..., platform as a service (PaaS)..., function as a service (FaaS), and other types of services such as desktop as a service (DaaS), information technology management as a service (ITaaS), managed software as a service (MSaaS), mobile backend as a service (MBaaS). etc.

Client endpoints 116 connect with the cloud 102 to obtain one or more specific services from the cloud 102. For example, the client endpoints 116 communicate with elements 104.-114 via one or more public networks (e.g., Internet), private networks, and/or hybrid networks (e.g., virtual private network).

Ex. 1007, [0022]-[0024].



Ex. 1007, FIG. 1.

148. Jeuk teaches that cloud environment 102 may include a cloud operator environment with a workload analysis engine 208 component that “analyze[s] the performance information received from the tenants 204, 206 of the cloud environment to identify performance patterns of the executing applications and/or correlate performance patterns across tenant configurations of the cloud,” that “correlate[s] each performance pattern generated for each tenant to a particular tenant configuration or deployment,” and provides the correlated pattern information to a recommendation engine 214 of the cloud operator. Ex. 1007, [0026], [0028], [0036], [0037]. The recommendation engine then “provide[s] a recommendation to one or more tenants...for improving the execution or deployment of workloads on the environment.” Thus in Jeuk, similarly as in Chernin, a centralized entity receives and analyzes information from respective entities, and information is then distributed to those entities for use in improving their performance—in the case of Jeuk, for executing or deploying workloads based on what is learned from other tenants, and in the case of Chernin, for addressing threats at other entities.

149. As I discussed above, a POSITA would have been motivated to combine Chernin and Jeuk, and applied Chernin’s teachings to cloud deployments (assuming Chernin itself does not teach using cloud deployments). Thus, it would have been obvious to deploy the “entities such as individuals, firms, government agencies and

associations, each of which are hosted on one or more distributed servers,” disclosed in Chernin, in the cloud, as Jeuk teaches.

150. Accordingly, the Chernin-Jeuk combination render obvious 1[pre]/9[pre]/17[pre].

1[a]/9[a]/17[a]: “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning;”

151. As I discussed above, in my opinion Chernin alone discloses or renders obvious this limitation.

152. But to the extent Chernin does not specifically disclose “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning,” a POSITA would have found it obvious to modify Chernin in view of Jeuk to do so.

153. Jeuk’s workload analysis engine “may identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants.” Ex. 1007, [0017]. In other words, Jeuk’s engine identifies which tenant deployments are running similar workloads—that is, it identifies one or more additional cloud deployments to utilize for cross-customer learning. Moreover, Jeuk teaches that the engine “correlate[s] each performance pattern generated for each tenant to a particular tenant configuration or deployment,” Ex.

1007, [0036], thereby associating the identified additional cloud deployments with their respective configurations so that the recommendation engine can generate targeted deployment recommendations.

154. As I discussed above, the examiner already found that Jeuk disclosed the concept of learning from cloud deployments during examiner of the '695 patent's parent. For example, as the examiner found, Jeuk teaches that a "workload engine may utilize the received performance information to detect performance patterns of an application across the different tenant deployments." Ex. 1007, Abstract, [0017] ("The workload engine may utilize the received performance information and one or more machine learning approaches to detect performance patterns of an application across the different tenant deployments. ***In other words, the engine may identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants.*** Once a similar application is identified as executed or deployed by different tenants on the cloud environment..."), [0031] ("Beginning in operation 402, the workload analysis engine 208 receives the log entries from one or more tenants of the cloud computing environment over a period of execution time. For example, tenant Y 204 may execute one or more applications in the tenant Y space of the cloud environment. The performance logs (listing a sequence of events for a particular application may

be transmitted to the workload analysis engine 208 by tenant Y 204 or the cloud environment 102. Similarly, tenant X 206 may execute one or more applications that may be the same or different than tenant Y applications) in the tenant X space 206 of the cloud environment 102.”).

155. As I discussed in the motivation to combine section, a POSITA would have been motivated to combine Jeuk and Shanks. In the combination, Jeuk’s engine would be deployed in Chernin’s central repository and would identify cloud deployments for cross-customer learning using the same analysis disclosed in Jeuk. *E.g.*, Ex. 1007, Abstract, [0017] (engine “may *identify a performance characteristic or pattern that may indicate the execution of a similar application by different cloud environment tenants*” and “a similar application is identified as executed or deployed by different tenants on the cloud environment...”). Thus, the system would correlate deployments using, for example, similar applications, which would improve information sharing in the event of an attack.

156. Accordingly, the Chernin-Jeuk combination renders obvious 1[a]/9[a]/17[a].

1[b]/9[b]/17[b]: “receiving information describing a security threat to one or more of the additional cloud deployments;”

157. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim limitation. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

1[c]/9[c]/17[c]: “receiving information describing configuration settings used to combat the security threat; and”

158. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim limitation. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

1[d]/9[d]/17[d]: “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.”

159. As I discussed above, in my opinion Chernin alone discloses or renders obvious this limitation.

160. But to the extent Chernin does not specifically disclose “identifying... configurations to adopt for the first cloud deployment,” a POSITA would have found it obvious to modify Chernin in view of Jeuk to do so.

161. As I discussed in the summary of Jeuk, Jeuk discloses a recommendation engine that correlates metrics and log information to generate deployment recommendations. Ex. 1007, [0018], [0037]. The recommendation engine analyzes the performance characteristics across multiple tenant applications and determines a

“best fit” or optimized deployment of the application. Ex. 1007, [0018]. Then, the recommendation engine provides the optimized deployment to a cloud environment administrator and/or one or more of the tenants. Ex. 1007, [0019].

162. During prosecution of the parent patent, the examiner found Jeuk discloses “identifying, based on the configurations, one or more configurations to adopt for the first cloud deployment.”

163. A POSITA would have been motivated to incorporate Jeuk’s correlation and recommendation engine into Chernin’s system for the reasons I discussed in the motivation to combine section.

4. Claims 2/10/18

2[pre]: “The method of claim 1” / 10[pre]: “The system of claim 9” / 18[pre]: “The computer program product of claim 17”

164. Claim 2 depends from claim 1; claim 10 depends from claim 9; and claim 18 depends from claim 17. The Chernin-Jeuk combination teaches claims 1/9/17.

2[a]/10[a]/18[a]: “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.”

165. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

5. Claims 3/11/19

3[pre]: “The method of claim 1” / 11[pre]: “The system of claim 9” / 19[pre]: “The computer program product of claim 17”

166. Claim 3 depends from claim 1; claim 11 depends from claim 9; and claim 19 depends from claim 17. The Chernin-Jeuk combination teaches claims 1/9/17.

3[a]/11[a]/19[a]: “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains.”

167. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

6. Claims 5/13

5[pre]: “The method of claim 1 further comprising” / 13[pre]: “The system of claim 9, wherein the steps further comprise”

168. Claim 5 depends from claim 1; and claim 13 depends from claim 9. Chernin and Jeuk teach or render obvious claims 1/9.

5[a]/13[a]: “receiving information describing one or more actions associated with the additional cloud deployments; and”

169. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim limitation. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

5[b]/13[b]: “wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions.”

170. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim limitation. By extension, Chernin likewise discloses or renders obvious this limitation when combined with Jeuk.

6. Claims 8/16

**8[pre]: “The method of claim 1 further comprising” /
16[pre]: “The system of claim 9 wherein the steps further comprise”**

171. Claim 8 depends from claim 1; and claim 16 depends from claim 9. The Chernin-Jeuk combination teaches claims 1/9.

8[a]/16[a]: “receiving information describing one or more deployment processes associated with the additional cloud deployments.”

172. As I discussed in Ground 1, Chernin alone discloses or renders obvious this claim limitation.

173. But to the extent Chernin alone does not disclose or render obvious this limitation, it would have been obvious in view of Jeuk.

174. During prosecution of the parent patent, the examiner found that Jeuk disclosed this limitation, as I described above. Specifically, the examiner reasoned that Jeuk met this limitation because it disclosed that “applications deployed and

executed on the cloud environment may provide performance logs and / or metrics to an inter - tenant workload engine of the cloud environment.”

175. Beyond what the examiner found, Jeuk provides additional direct support. Jeuk teaches that the recommendation engine “may receive or obtain[] deployment information for the workloads or applications related to the identified performance patterns” and that “[s]uch deployment information may be any type of information related to the deployment of the applications or workloads, including but not limited to, devices utilized, physical and/or virtual location of resources, whether resources are shared or dedicated, particular standards or protocols utilized, etc.” Ex. 1007, [0043].

176. Accordingly, the Chernin-Jeuk combination renders obvious 8[a]/16[a].

VIII. GROUND 3: THE COMBINATION OF HUDIS AND SHANKS RENDERS CLAIMS 1-3, 5, 8-11, 13, 16-19 OBVIOUS.

177. In my opinion, the combination of Hudis and Shanks renders claims 1-3, 5, 8-11, 13, 16-19 obvious.

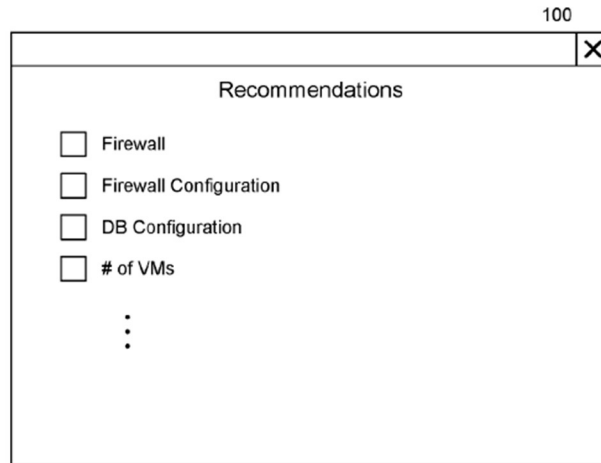
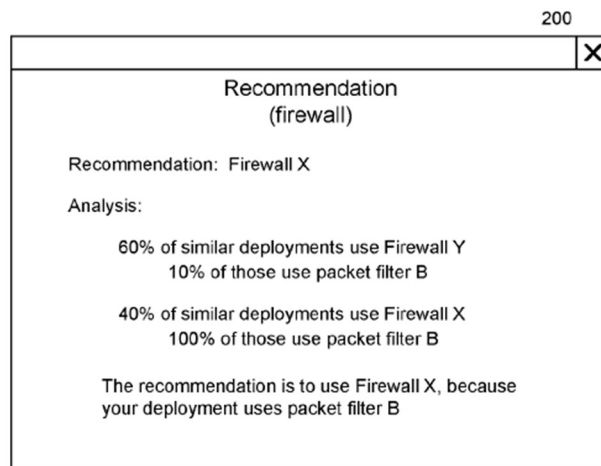
A. Technical Overview of Hudis

178. Hudis discloses a recommendation system that recommends feature values for a target deployment within a data center by analyzing the feature values used by other deployments. Ex. 1008, Abstract, [0019]. Hudis’s system recommends a feature value for a target feature based on the similarity between the features of the target

deployment and those of other deployments in the data center. *Id.*, [0019]. To give a concrete example, “if . . . features of the target deployment are most similar to features of deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name.” *Id.*, [0020].

179. Hudis provides exemplary display pages showing (1) how a user can select which recommendations it wants to receive and (2) a recommendation for a firewall based on which firewalls other, similar deployments are using.

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

**FIG. 1****FIG. 2**

Ex. 1008, Figs. 1 and 2.

180. To generate its recommendations, the recommendation system collects feature values for each deployment within the data center. *Id.*, [0019]. These features may relate to data center resources (e.g., virtual machines and storage), appliances (e.g., firewalls and packet filters), configuration information and log files of data center

resources and appliances, usage patterns, network data, and indicators of compromise regarding attacks and threat intelligence. *Id.*, [0019], [0040]. The system also collects threat intelligence that, for example, “identifies Internet Protocol (‘IP’) addresses whose access should be blocked by a firewall.” *Id.*, [0040].

181. Rather than generating a single model across all deployments, the system can generate separate models for clusters of similar deployments. *Id.*, [0022]. The system clusters the deployments based on similarity between features of the deployments, for example using a k-means or k-nearest neighbor clustering algorithm. *Id.* These are standard machine learning techniques that any competent systems engineer would have been familiar with by 2017. The system then generates a model for each cluster using training data from the deployments within that cluster. *Id.* To generate a recommendation, the system first identifies the cluster to which the target deployment is most similar. *Id.*, [0023]. The system determines similarity, for example, by computing cosine similarity between a representative feature vector for each cluster and the feature vector for the target deployment. *Id.* After identifying the most similar cluster, the system applies the model for that cluster to the target deployment’s feature vector to generate the recommendation. *Id.*

182. Hudis further discloses that the recommendation system can monitor the features of deployments in real time and provide recommendations based on changes

in those features. *Id.*, [0024]. From an engineering standpoint, this is precisely the kind of feedback loop one would design into a security system. Hudis describes a scenario in which “several deployments are the subject of a certain type of attack” and “some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack.” *Id.* The recommendation system can use such changes as a trigger to generate recommendations for changing the configurations of firewalls of other deployments. *Id.* The trigger can be based on a recommendation criterion such as the percentage of deployments that change their configuration in a certain time period, the sophistication of the deployments, or the reputation of administrators of the deployments. *Id.* The system then “generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.” *Id.*

183. Hudis also teaches that a deployment may authorize the automatic implementation of recommendations. *Id.* at [0024]. In that case, “the recommendation system may direct a configuration manager of the deployment to change the configuration to automatically implement the recommended configuration without the explicit approval of an administrator of the deployment.” *Id.* Hudis thus provides a mechanism by which firewall configuration settings that have proven effective at thwarting attacks on some deployments can be identified

and automatically propagated to other similar deployments within the data center. *Id.*, [0024]-[0025]. In my professional judgment, this is a straightforward application of automated configuration management, a technique that was already well established in the field by 2017.

B. Technical Overview of Shanks

184. Shanks relates to automatically configuring firewalls to minimize errors caused when administrators manually configure their firewalls. Ex. 1009, [0002]. As someone working in the area of computer security, I immediately recognize this as addressing a well-known pain point: manual firewall configuration is notoriously error-prone, and automated approaches to reduce human error were actively sought throughout the industry. Shanks's architecture is geared towards firewall appliance providers, which makes its practical applicability all the more evident. Indeed, the assignee of the Shanks patent is Cisco, a major player in the network security space that manufactures and sells firewall appliances. Shanks also discloses that customers of a cloud management system could "purchasing a specified service" to have their firewalls configured automatically, which would be an attractive additional recurring revenue stream for a manufacturer of firewall appliances. *Id.* at [0068].

185. Shanks discloses systems, methods, and computer-readable storage media for "automatic firewall configuration based on aggregated cloud managed information"

using cross-customer learning. Ex. 1009, [0011]. A cloud management system (which can itself be cloud-based) communicates with multiple client computing environments that may belong to different customers. *Id.*, [0061], [0068]. Client computing environments can continuously transmit “security event data” to the cloud management system. *Id.*, [0060], [0063]. This architecture is depicted in Figures 4 and 5, below.

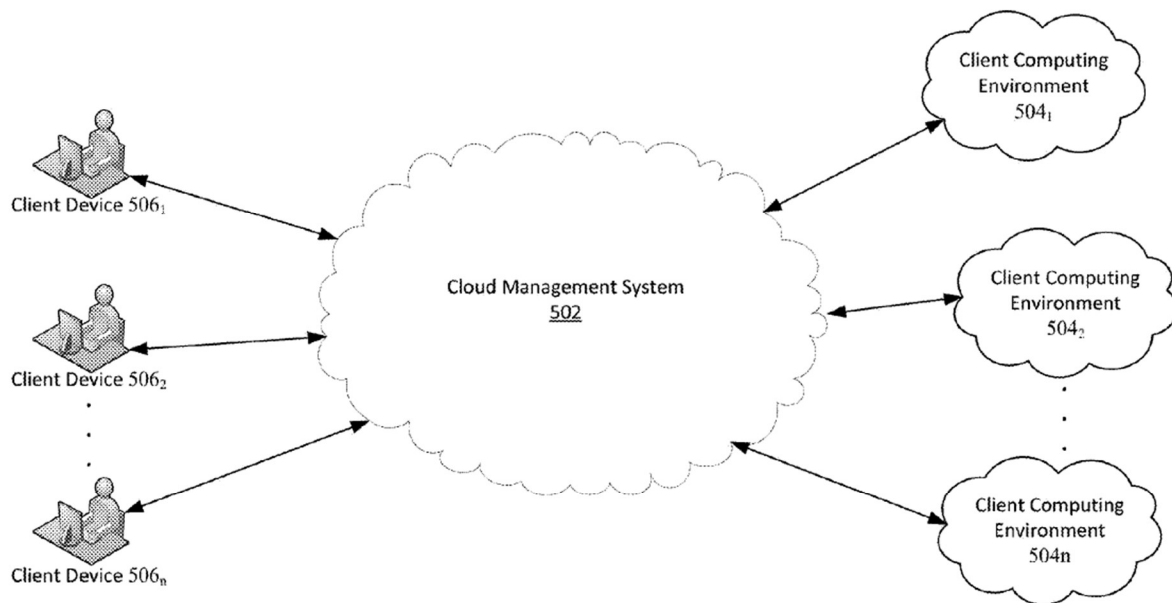
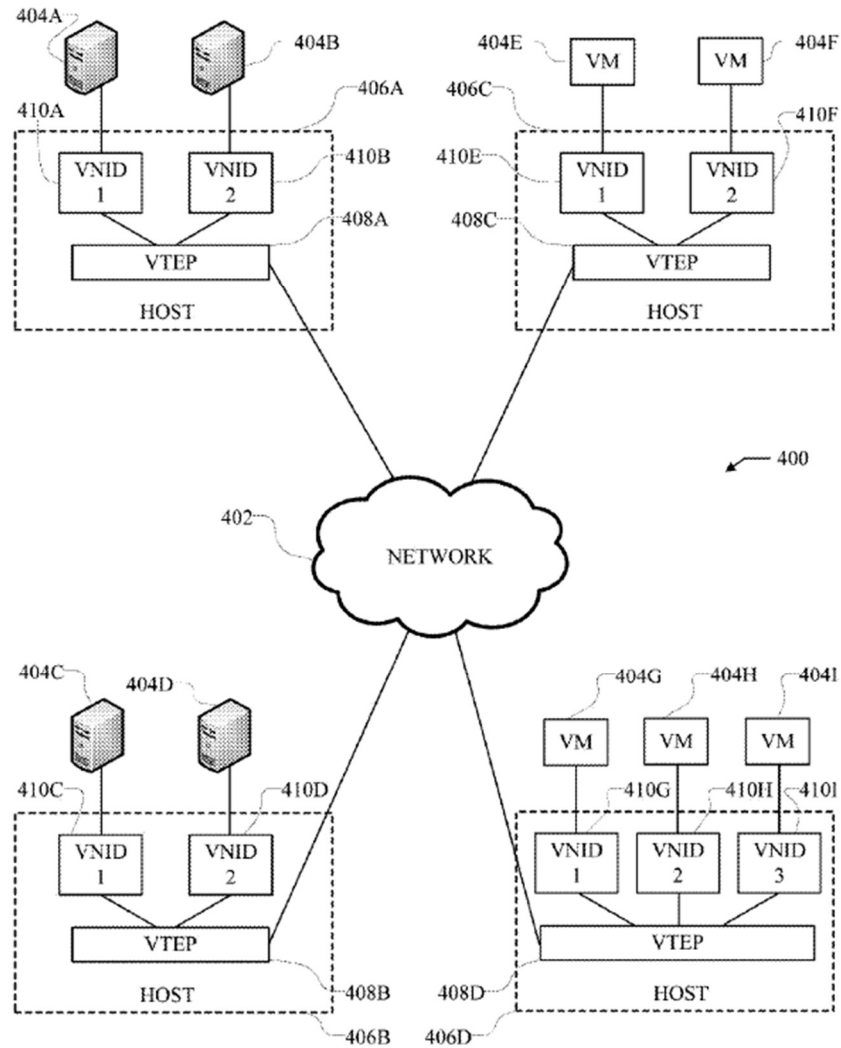
**FIG. 5**

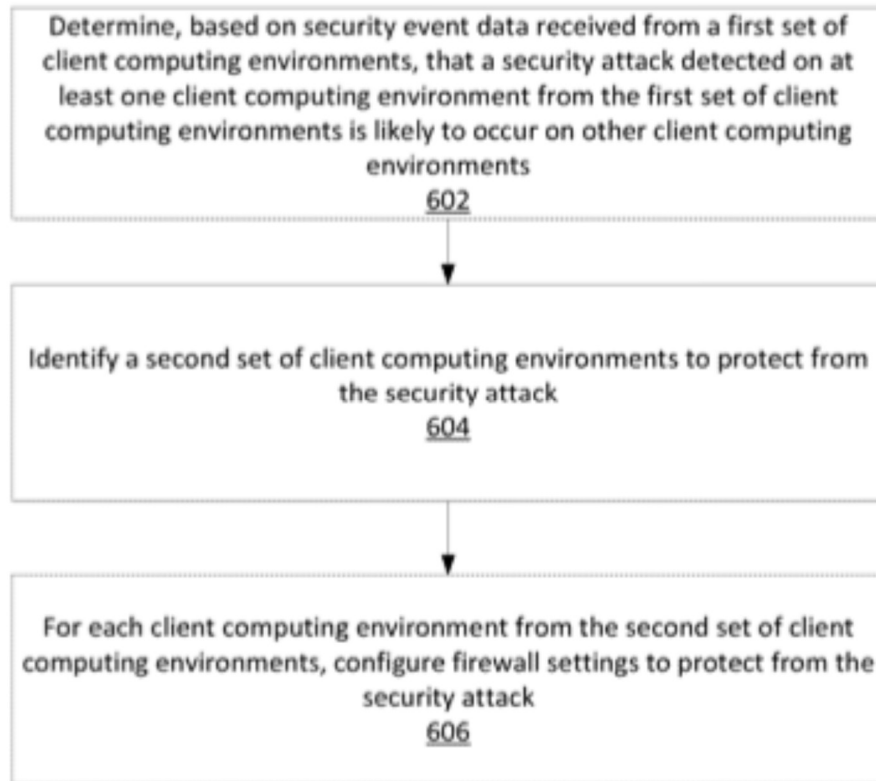
FIG. 4



Ex. 1008, Figs. 4 and 5.

186. Security event data can include any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device. *Id.*, [0063]. A security event can be any type of observable change to the normal behavior of a computing system,

device, environment, process, workflow, etc., that can indicate a malicious security breach. *Id.*



Ex. 1009, Fig. 6

187. The cloud management system can analyze such security event data received from a first set of client computing environments to identify security attacks and determine whether the security attacks are likely to occur on other client computing environments. *Id.*, at [0060], [0064]. In response to determining that a detected security attack is likely to occur on other client computing environments, the cloud management system identify a second set of client computing environments to

protect from the security attack. *Id.*, [0060], [0073]. For each of the client computing environments from this second set, it can then automatically configure firewall settings. *Id.*, [0068], [0074]. For example, it can apply firewall rules to block and/or filter data packets transmitted from a source associated with the security attack or to block and/or filter data packets transmitted to a destination port associated with the security attack. *Id.*, [0069].

188. Shank's cloud management system can provide users with a management interface that allows users to set configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment. *Id.*, [0059], [0062]. Users (e.g., system administrators) communicate with the cloud management system to access the management interface and set or adjust configurations. *Id.*, [0062]. These received configurations can then be then sent to client computing environments for implementation. *Id.*, at [0059], [0062].

C. Motivation to Combine Hudis and Shanks

189. In my opinion, a POSITA at the time of the alleged invention would have been motivated to combine the disclosures of Hudis and Shanks and had a reasonable expectation of success in doing so.

190. Hudis and Shanks are both directed to cloud-based systems that aggregate data across cloud deployments, and use that data to improve performance of other cloud

deployments. Hudis aggregates data across cloud deployments and uses that data to generate a recommendation on configurations for “data center resources (e.g., virtual machines and storage), appliances (e.g., firewalls and packet filters), configuration information and log files of the data center resources and appliances, and so on.” Ex. 1008, [0019]. Hudis also applies to recommending firewall configuration settings to thwart a security attack. *Id.*, [0024] (“In some embodiments, the recommendation system may monitor the features of the deployments in real time and provide recommendations based on changes in the features of the deployments. For example, if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack. The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments.”). Similarly, Shanks aggregates data across cloud deployments about security events and uses that data to generate a configuration setting that can be used to thwart an attack in another deployment. Ex. 1009, [0011] (“Disclosed are systems, methods, and computer readable storage media for automatic firewall configuration based on aggregated cloud managed information.”). Thus, Hudis and Shanks are in the same field of

endeavor and address the same problem, providing a strong foundation to combine the two.

191. Moreover, a POSITA would have been motivated to combine the teachings of these references because doing so would have created a materially improved system that leverages the respective strengths of each.

192. As noted in the summary of Hudis, Hudis teaches collecting from cloud deployments configuration settings used to thwart a security attack. Ex. 1008, at [0024] (“For example, if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack. The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments.”). In this scenario, though, a recommended configuration setting is community-driven, i.e., derived from statistical analysis of the configuration settings the deployments implemented to thwart the attack. While that procedure can generate recommendations that would thwart the attack, it might not, or it might not necessarily recommend the optimal solution—for instance, the most effective countermeasure may have been less widely implemented simply because fewer administrators were aware of it. Furthermore, this approach leaves open the risk that

the administrator of the cloud deployment introduces errors when manually configuring the firewall, a problem Shanks specifically identifies and seeks to solve.

193. A more streamlined and reliable solution would be to have a cloud security management system recommend a solution, or even automatically configure the firewall itself. This is precisely what is taught by Shanks, except that Shanks uses the term “cloud management system” when it really talks about a cloud *security* management system. Such a cloud security management system, especially one provided by a firewall appliance vendor, is in the best position to (1) identify the optimal configuration setting to thwart a given attack, because it has deep knowledge of its own product’s capabilities and known vulnerabilities, and (2) deploy that setting rapidly across its customer base—both to existing customers and to new installations.

194. Hudis itself suggests this solution. Hudis contemplates that its invention is not limited to the specifically-disclosed implementations (Ex. 1008, [0049] (“Although the subject matter has been described in language specific to structural features and/or acts, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or acts described above. Rather, the specific features and acts described above are disclosed as example forms of implementing the claims.”))), and that, for example, a provider of a security

appliance (e.g., a firewall) can request that customers send feature value recommendations to them in order to make recommendations to current or new customers:

As an example, provider of security appliances may request that its customers provide their feature values of the security features of their installations of the security appliances. ***The provider can then, based on those feature values, make recommendations to its current customers for changes in feature values, new customers for initial feature values, and so on.***

Id., [0049]

195. Instead of relying solely on Hudis's community-driven recommendation, a POSITA would have been motivated to combine the references' teachings so that similar cloud deployments are clustered (as Hudis teaches), information is gathered from cloud deployments (as both Hudis and Shanks teach), information—especially about an ongoing attack—is shared with the security appliance provider (as Hudis teaches), a targeted configuration setting to thwart the attack is identified by the appliance provider (as Shanks teaches), and then the configuration setting is automatically implemented into the target deployment (as both Shanks and Hudis teach). This is a natural engineering design that follows directly from the respective capabilities of the two systems.

196. As mentioned in §179 above, Shanks is assigned to Cisco Technology, Inc.—one of the world's largest providers of network security appliances, including

firewalls. Shanks itself teaches that customers can opt in for automatic firewall configuration (Ex. 1009, [0068]), which is precisely the type of managed security service a major network security appliance provider such as Cisco would offer to its customers. In my experience, appliance vendors have long sought to differentiate their products through exactly this kind of automated, intelligence-driven configuration capability. A POSITA would have recognized that Shanks's automatic firewall configuration approach represents the kind of security-specific enhancement that a firewall vendor would incorporate into a broader cross-deployment recommendation platform like the one Hudis discloses.

197. The technical compatibility between the two systems is high, since both operate as centralized cloud management platforms collecting data from distributed deployments. Incorporating Shanks's automatic firewall learning into Hudis's general recommendation system represents an additive, predictable, and low-risk enhancement. From an engineering perspective, this is a straightforward case of combining known, well-tested elements to yield predictable results—no new protocols, no new data formats, and no fundamental architectural changes are required. Indeed, as I walk through how the combination renders the claims obvious below, I have identified overlapping and analogous disclosures across the references

that further support my opinion that it would be straightforward for an engineer of ordinary skill to integrate their respective teachings.

D. Claims 1, 9, and 17

1[pre]: “A method of learning from other cloud deployments to combat security threats, the method comprising”

9[pre]: “A system for learning from other cloud deployments to combat security threats, the system comprising at least one processor and memory storing computer program instructions that, when executed, cause the system to carry out the steps of”

17[pre]: “A computer program product for learning from other cloud deployments to combat security threats, the computer program product disposed on a non-transitory computer readable medium, the computer program product including computer program instructions that, when executed, carry out the steps of”

198. To the extent 1[pre]/9[pre]/17[pre] are limiting, the combination of Hudis and Shanks teach methods, systems, and computer program products for “learning from other cloud deployments to combat security threats,” as well as for performing such learning.

199. Hudis teaches learning from other cloud deployments to combat security threats. For example, Hudis teaches that a “target deployment” learns from configuration settings (e.g., firewall configuration settings) used by other cloud deployments to thwart an attack.

200. Hudis discloses a “recommendation system for recommending a feature value for a target feature for a target deployment within a data center....” Ex. 1008, [0019]. Hudis’s invention also applies to “multiple data centers that may include deployments in private data centers of organizations, deployments in cloud-based data centers, and so.” *Id.*, [0049].

201. Hudis “recommends a feature value for the target feature based on similarity between the features of the target deployment and the features of other deployments in the data center.” Hudis gives an example of sharing a feature value in the context of a firewall. In this example, Hudis explains that “if features of the target deployment are most similar to features of deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name.” *Id.*, [0020].

202. This is reflected in Figures 1 and 2 (below). Figure 1 shows an interface wherein an administrator can select the features for which it wants recommendations (e.g., “firewall,” “firewall configuration,” “DB configuration”). Figure 2 shows an example of a firewall recommendation based on firewalls used in similar deployments:

100

×

Recommendations

☐ Firewall

☐ Firewall Configuration

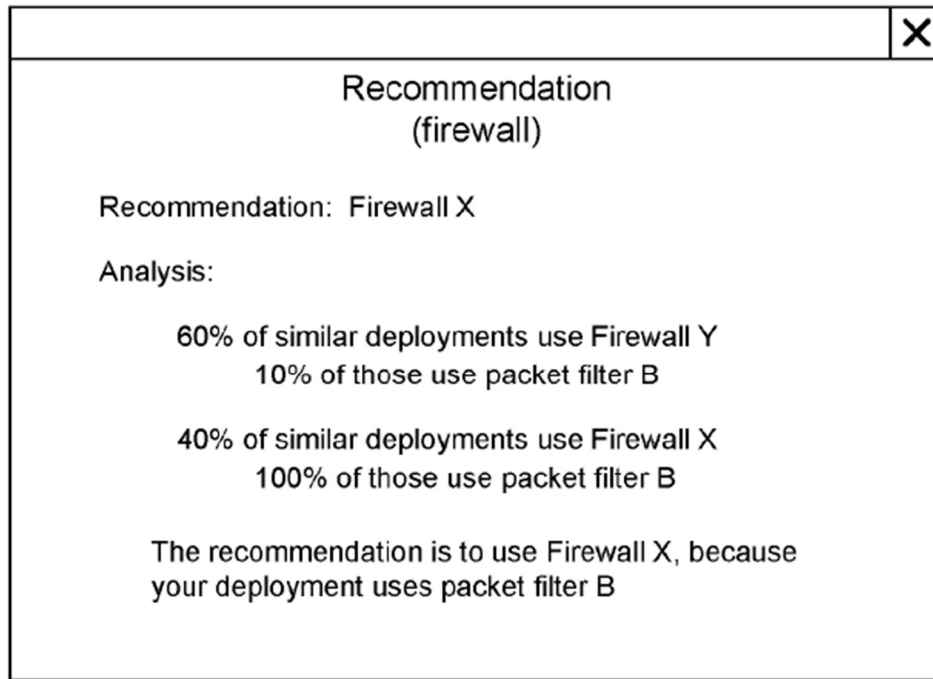
☐ DB Configuration

☐ # of VMs

⋮

FIG. 1

200

**FIG. 2**

Ex. 1008. Figs. 1 and 2.

203. Hudis' learning from other cloud deployments can be used to combat security threats in real time.

For example, *if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack. The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments.* The trigger may be based on a recommendation criterion such as a percentage of deployments that change their configuration in a certain time period, the sophistication of the deployments, the

reputation of administrators of the deployments, and so on.

The recommendation system generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.

Id. at [0024]; [0025] (“For example, a target deployment may use a default configuration for a firewall that is not effective at thwarting certain types of attacks. As other deployments within the data center change their configurations to thwart an attack, the recommendation system may direct the target deployment to change its configuration accordingly.”).

204. The recommendation system can automatically implement recommended configurations for the target deployment. *Id.* (“In some embodiments, a deployment may authorize the automatic implementing of recommendations. In such a case, the recommendation system may direct a configuration manager of the deployment to change the configuration to automatically implement the recommended configuration without the explicit approval of an administrator of the deployment.”)

205. Shanks is consistent with Hudis. Shanks teaches a cloud management system that receives security event data (security threats) from CCEs and analyzes this aggregated information to determine if security attacks are likely to occur on other CCEs (learning from other cloud deployments) so as to combat the threats.

206. For example, Shanks discloses:

systems, methods, and computer-readable storage media for automatic firewall configuration based on aggregated cloud managed information. A cloud management device can determine, based on security event data received from a first set of client computing environments, that a security attack detected on at least one client computing environment from the first set of client computing environments is likely to occur on other client computing environments. **In response to determining that the security attack detected on at least one client computing environment from the first set of client computing environments is likely to occur on other client computing environments, the cloud management device can identify a second set of client computing environments to protect from the security attack. For each client computing environment from the second set of client computing environments, the cloud management device can configure firewall settings to protect from the security attack.**

Ex. 1009, Abstract. Shanks further discloses that a **“cloud management system can allow users to manage client computing environments,”** for example providing a **“management interface that allows users to set configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment”** *Id.*, [0059]. Shanks further teaches with regards to FIG. 5, reproduced below, that “[c]loud management system 502 can be a cloud based computing system comprising one or more computing device (e.g., cloud management devices) that allows users to remotely manage their client computing

environments.” Ex. 1009, [0061]. Shanks further teaches that a CCE “can include multiple computing nodes, such as firewalls, servers, routers, access points, etc.” *Id.*

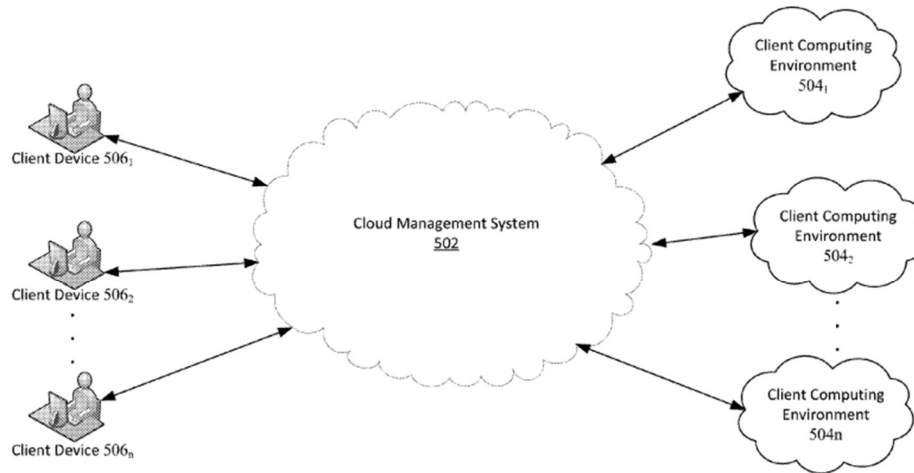


FIG. 5

207. In these excerpts, Shanks discloses various CCEs that are managed by a “cloud management device” or a “cloud management system.” Shanks’s CCEs are “cloud deployments,” and indeed FIG. 5 of Shanks even uses cloud icons to illustrate the CCEs. *See* Ex. 1009, [0021], [0061], FIG. 5. Moreover, Shanks’s “other” CCEs correspond to the “other cloud deployments.”

208. Shanks teaches that the cloud management system learns from the other CCEs (that is, learns from “other cloud deployments”) to combat security threats.

Illustratively, Shanks discloses that:

client computing environments can continuously transmit security event data to the cloud management system. The cloud management system can analyze the security event data to identify security attacks and determine whether the

security attacks are likely to occur on other client computing environments. **In response to determining that a detected security attack is likely to occur on other client computing environments, the cloud management system can automatically configure firewall setting on client computing environments to protect from the security attack.**

Ex. 1009, [0060]. Shanks further discloses that “[a] security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach.”

Id., [0063]. Shanks’s disclosure of identifying security attacks/security events, by analyzing security event data that the client computing networks continuously transmit to the cloud management system, corresponds precisely to “learning from other cloud deployments to combat security threats” as recited in 1[pre]/9[pre]/17[pre].

209. Hudis and Shanks teach performing operations as part of a “method,” as in 1[pre] and a “system” as in 9[pre]. *See, e.g.*, Ex. 1009, Abstract.

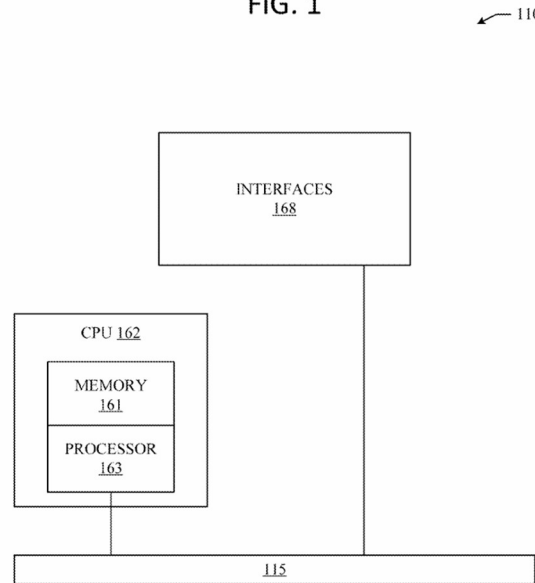
210. Hudis teaches that the recommendation system can be executed on a computing system that “may include a central processing unit, input devices, output devices (e.g., display devices and speakers), storage devices (e.g., memory and disk drives), network interfaces, graphics processing units, accelerometers, cellular radio link interfaces, global positioning system devices, and so on.” Ex. 1008, [0033].

Hudis goes on to teach that “[t]he computing systems may access computer-readable media that include computer-readable storage media and data transmission media.” *Id.*; *see also id.*, [0045] (“The processing described below may be performed by a computing device with a processor that executes computer-executable instructions stored on a computer-readable storage medium that implements the recommendation system.”); [0047] (“The computing system comprises a processor for executing computer-executable instructions and a computer-readable storage medium storing computer-executable instructions.”); claim 11. The recommendation system can be implemented as a program. *Id.*, [0034] (“The recommendation system may be described in the general context of computer-executable instructions, such as program modules and components, executed by one or more computers, processors, or other devices.”).

211. Shanks teaches that the system can include a processor and memory storing computer program instructions that, when executed, cause the system to carry out the disclosed steps. Illustratively, Shanks discloses with reference to FIG. 1, reproduced below, that the “present technology” may be implemented using a network device 110 including a master central processing unit (CPU) 162 that “may include one or more processors” and a “memory 161 (such as non-volatile RAM and/or ROM).” *See Ex. 1009*, [0023], [0078]. Shanks teaches that the memory stores “instructions

that, when executed by the one or more computer processors, cause the cloud management system” to perform the steps disclosed by Shanks. *Id.*, claim 8.

FIG. 1



Ex. 1009, FIG. 1.

212. Shanks also discloses non-transitory “computer-readable storage media” including computer-executable instructions that, when run by a cloud management system, perform the recited steps. For example, Shanks teaches:

Methods according to the above-described examples can be implemented using **computer-executable instructions that are stored or otherwise available from computer readable media**. Such instructions can comprise, for example, instructions and data which cause or otherwise configure a general purpose computer, special purpose computer, or special purpose processing device to perform a certain function or group of functions. Portions of computer resources used can be accessible over a network.

The computer executable instructions may be, for example, binaries, intermediate format instructions such as assembly language, firmware, or source code. Examples of computer-readable media that may be used to store instructions, information used, and/or information created during methods according to described examples include magnetic or optical disks, flash memory, USB devices provided with non-volatile memory, networked storage devices, and so on.

Ex. 1009, [0078]; *see also id.*, [0083]-[0084].

213. Accordingly, to the extent that the 1[pre]/9[pre]/17[pre] are limiting, the Hudis-Shanks combination teaches them.

1[a]/9[a]/17[a]: “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning;”

214. The Hudis-Shanks combination teaches “identifying, for at least a portion of a first cloud deployment, one or more additional cloud deployments to utilize for cross-customer learning,” as recited in 1[a]/9[a]/17[a].

215. Hudis teaches a recommendation system that recommends a feature value for a target feature for a target deployment (“first cloud deployment”) by learning and analyzing feature values for the target feature in other deployments (“identifying...one or more additional cloud deployments to utilize for cross-customer learning”).

216. Hudis “recommends a feature value for the target feature based on similarity between the features of the target deployment and the features of other deployments....” Ex. 1008, [0019]. To generate the recommendation, the system collects feature values for features, including a target feature, from cloud deployments. *Id.* Then, the system generates a model using the collected feature values for recommending a feature value for a target feature for a target deployment. *Id.*, [0020]. Once the model is generated, the “system applies the model to the features of the target deployment to identify a feature value for the target feature and provides the identified feature value as a recommendation for the target feature for the target deployment.” *Id.* Hudis provides an example of how this cross-deployment learning process works:

For example, if the features of the target deployment are most similar to features of deployments that use a firewall with a certain name, then the recommendation system recommends that the target deployment use the firewall with that certain name. The recommendation system thus performs collaborative filtering to base recommendations on deployments with similar features.

Ex. 1008, [0020].

217. Hudis teaches that, instead of generating a model for each feature using the features of all deployments as training data, the system can generate “separate models for clusters of similar deployments.” *Id.*, [0021]-[0022]. In this arrangement, the system “clusters the deployments based on similarity between features of the

deployments.” *Id.*, at [0022.] For example, similarity can be determined based on a k-means clustering algorithm or a k-nearest neighbor clustering algorithm. *Id.*

218. The system can select a subset of features for generating the clusters and generate a model for each cluster using training data from each deployment (e.g., “a feature vector of feature values of features of that deployment that is labeled with the feature value of the target feature (i.e., target feature value) of that deployment”).

Id., [0022]. Hudis discloses flexibility in how the clusters and models are generated.

For example, the system can use different feature sets when generating a cluster versus the model, or can even use different feature sets when generating the models.

Id. Hudis provides an example of when this might occur:

For example, it may be that all the deployments in a certain cluster have the same feature value for a certain feature. In such a case, that certain feature is not informative to distinguish feature values for the target feature. Another cluster, however, may have different feature values for that certain feature and thus may be informative to distinguish feature values for the target feature.

Id.

219. Additionally, Hudis discloses that the system can first identify a cluster of deployments “to which the target deployment is most similar.” *Id.*, [0023]. In this situation, Hudis discloses generating a representative feature vector for each cluster.

Id. The representative feature value can be an average, mean, or median of the

feature values within the cluster (e.g., number of virtual machines) or it can indicate the percentage of deployments in a cluster that have a representative feature (e.g., name of a firewall) for non-numerical features. *Id.* Hudis explains that the system identifies “the most similar cluster based on cosine similarity between the representative feature vector for a cluster and the feature vector for the target deployment.” *Id.* After the most similar cluster is identified, the “system applies the model for that cluster to the feature vector of the target deployment to generate the recommendation for the target feature.” *Id.*

220. Hudis teaches applying this cross-deployment learning in the context of thwarting security attacks:

For example, if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack. The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments. The trigger may be based on a recommendation criterion such as a percentage of deployments that change their configuration in a certain time period, the sophistication of the deployments, the reputation of administrators of the deployments, and so on. *The recommendation system generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.*

Id., [0024]; [0025] (“For example, a target deployment may use a default configuration for a firewall that is not effective at thwarting certain types of attacks. As other deployments within the data center change their configurations to thwart an attack, the recommendation system may direct the target deployment to change its configuration accordingly.”).

221. Shanks’s cloud management system is consistent with Hudis’.

222. Shanks teaches that the cloud management system identifies, for at least a portion of a CCE, one or more additional CCEs to utilize for cross-customer learning.

For example, Shanks teaches the following with reference to step 602 of FIG. 6, reproduced further below:

At step 602, a cloud management device can determine, based on security event data received from a first set of client computing environments, that a security attack detected on at least one client computing environment from the first set of client computing environments is likely to occur on other client computing environments. **The cloud management system can receive, from the first set of client computing environments, security event data that identifies security events detected on client computing environments from the first set of client computing environments.**

Ex. 1009, [0071].

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

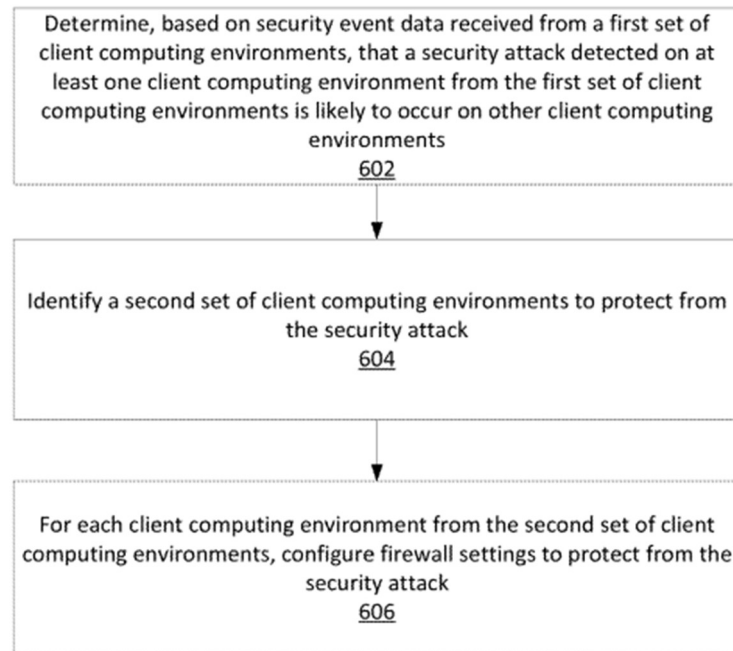


FIG. 6

Ex. 1009, Fig. 6.

223. Moreover, Shanks teaches, again with reference to FIG. 5 further above, that:

In addition to enabling users to easily manage their client computing environments 504, cloud management system 502 can also automatically configure firewalls based on aggregated cloud managed information. **For example, client computing environments 506 can continuously transmit security event data to cloud management system 502. Security event data can include any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device.** A security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach.

Ex. 1009, [0063]. Here, Shanks teaches that CCEs (“cloud deployments”) continuously transmit, to cloud management system 502, security data describing security events that can indicate a malicious security breach.

224. Shanks further teaches that cloud management system 502 analyzes the security data describing security events that can indicate a malicious security breach for one of the CCEs (“cloud deployments”), to identify other CCEs vulnerable to a similar breach. For example, Shanks teaches that based on analyzing the security event data, e.g., as in step 602 of FIG. 6:

At step 604, the cloud management device can identify a second set of client computing environments to protect from the security attack. For example, the cloud management system can identify client computing environments that have opted in to automatic firewall configurations. It should be noted that there can be overlap between the first and second set of client computing environments.

Ex. 1009, [0073].

225. And, with reference to FIG. 5 above, Shanks further teaches that:

Cloud management system 502 can analyze the security event data to identify security attacks that have occurred on client computing environments 506 and determine whether the security attacks are likely to occur on other client computing environments 506. Cloud management system 502 can determine whether a security attack is likely to occur on other computing devices in multiple ways and based on multiple factors.

For example, cloud management system 502 can analyze vulnerability scores associated with detected security attacks to determine whether one or more of the vulnerability scores meets or exceeds threshold vulnerability score. Vulnerability scores can be predetermined scores set by an administrator or other user associated with cloud management system 502. Alternatively, vulnerability scores can be determined based on industry standards, such as vulnerability scores set by Common Vulnerabilities and Exposures (CVE). Cloud management system 502 can determine that the security attack is likely to occur on other client computing environments 506 if one or more of the vulnerability scores does meet or exceed the threshold vulnerability score.

Ex. 1009, [0064].

226. Accordingly, Shanks teaches identifying security attacks on CCEs (“cloud deployments”) and determining whether the security attacks are likely to occur on other CCEs.

227. In these teachings, Shanks further teaches identifying a second set of CCEs to protect from the security attack, using security event data from a first CCE. The use of that security event data for this purpose corresponds to “cross-customer learning.” Indeed, Shanks discloses that “customers (e.g., businesses, schools, government institutions, etc.) that utilize cloud management system 502 to manage their client computing environment may select to have their firewalls automatically configured to protect from likely security threats.” Ex. 1009, [0068]. Where customers opt-in for the cloud management system 502 to manage their firewall configuration, such

customers opt-in for “cross-customer learning” where the cloud management system configures the firewall based on detected security threats to other customers.

228. Accordingly, the Hudis-Shanks combination teaches 1[a]/9[a]/17[a].

1[b]/9[b]/17[b]: “receiving information describing a security threat to one or more of the additional cloud deployments;”

229. The Hudis-Shanks combination teaches “receiving information describing a security threat to one or more of the additional cloud deployments,” as in 1[b]/9[b]/17[b].

230. The ’695 patent provides the following examples of “information describing a security threat to one or more of the additional cloud deployments:”

The information describing a security threat to one or more of the additional cloud deployments 912 can include, for example, information describing where an attack originated from, information describing the components within the cloud deployment that were targeted by an attack, information describing data access patterns that were associated with an attack, or some other information that would be useful in detecting subsequent security threats.

Ex. 1001, 81:31-38.

231. Hudis teaches that “if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack.” Ex. 1008, [0024]; [0025] (“As other deployments within the data center change their configurations to thwart an

attack, the recommendation system may direct the target deployment to change its configuration accordingly.”). The system can use those changes to generate a recommendation for changing firewall configurations for the target deployment. *Id.*

232. While Hudis does not expressly disclose receiving information “describing” a security threat, that feature is suggested by Hudis. Hudis discloses generating recommendations with configuration settings to thwart an attack. Hudis at [0024]-[0025]. Hudis also discloses collecting “various indicators of compromise regarding attacks and threat intelligence from the deployments” and IP addresses that should be blocked. *Id.* at [0040] (“In block 907, the component collects various indicators of compromise regarding attacks and threat intelligence from the deployment and then completes. The component may, for example, analyze usage patterns to determine whether an attack (e.g., denial of service) was attempted. A deployment may purchase threat intelligence that, for example, identifies Internet Protocol (“IP”) addresses whose access should be blocked by a firewall.”). Thus, Hudis suggests collecting information describing an attack (e.g., indicators of compromise, threat intelligence) from cloud deployments. Indeed, such information would have been important to generating a recommendation for thwarting the attack (e.g., one would need to know the IP address to block before being able to recommend blocking it).

233. Even if Hudis does not expressly teach or suggest this element, it would have been obvious in light of Hudis' own teachings and Shanks. As noted above, Hudis collects indicators of compromise and threat intelligence from cloud deployments.

234. Additionally, Shanks teaches that CCEs continuously transmit security event data to the cloud management system; thus the cloud management system receives such data. Shanks's received security event data corresponds to "information describing a security threat to one or more of the additional cloud deployments."

235. Shanks similarly teaches the received security event data can include:

any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device. A security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach.

Ex. 1009, [0063].

236. Shanks discloses that the CMS can apply a rule to thwart the attack in other deployments. For example, the CMS can block "a source associated with the attack" or "destination port associated with the security attack." Ex. 1009, [0069]. To apply these rules, the CMS would have needed the identity of the malicious source or malicious destination, i.e., information describing the attack. The most obvious place

to collect this information is from the deployments that already experienced the attack.

237. Shanks's "data describing security events detected in client computing devices" corresponds precisely to "information describing a security threat to one or more of the additional cloud deployments" as in 1[b]/9[b]/17[b].

238. A POSITA would have been motivated to combine the teachings of Hudis and Shanks for the reasons I discussed above. In the combination, Shanks' teachings about data describing security events would be part of the information that Hudis' recommendation system collects from each cloud deployment. Indeed, Hudis already discloses collecting indicators of compromise, threat intelligence, and blacklisted IP addresses. A POSITA would have been motivated to collect that information in real time in the event of an attack because it would be most relevant to the system at that time when it could assist other deployments in thwarting the attack. Indeed, before the '695 patent, it was well-known and understood in the threat intelligence community to share information about an attack with the hope that the information may help another member of the community prevent the attack on them. Traditionally, relevant information to share may include a malicious IP address, file hash, domain, or abnormal login. Indeed, threat information sharing was so widespread before the '695 patent that the STIX and TAXII standards were

developed. As I discussed in the technology background, STIX standardized the format and type of information that would be shared in the event of an attack, including information describing the attack. Accordingly, it would have been obvious to a POSITA that the recommendation system—and, ultimately, any cloud [security] management system, for example one operated by an appliance provider, and target deployment—would receive information about the attack (e.g., source IP address, name of firewall targeted, access pattern is consistent with a DDOS attack) to increase the chances of thwarting the attack for others.

239. Accordingly, the Hudis-Shanks combination teaches 1[b]/9[b]/17[b].

1[c]/9[c]/17[c]: “receiving information describing configuration settings used to combat the security threat; and”

240. The Hudis-Shanks combination teaches “receiving information describing configuration settings used to combat the security threat,” as recited in 1[c]/9[c]/17[c].

241. The '695 patent states:

The configuration settings used to combat the security threat may be embodied, for example, as configuration settings that caused the additional cloud deployments 912 to blacklist certain IP addresses or network domains, as configuration settings that caused the additional cloud deployments 912 to close vulnerabilities that allowed an attack to succeed, or some other configuration that would

be useful in detecting/preventing/mitigating a security threat.

Ex. 1001, 81:43-51.

242. Hudis teaches that deployments have “various parameters or configuration settings” that are part of the deployment’s configuration, including parameters for security sub-systems such as “a firewall appliance, an anti-virus appliance, an encryption appliance, and so on.” Ex. 1008, [0002]. Hudis further teaches that features of deployments “may relate to data center resources (e.g., virtual machines and storage), appliances (e.g., firewalls and packet filters), configuration information and log files of the data center resources and appliances, and so on.” *Id.*, [0019].

243. Hudis discloses administrators changing firewall configurations to thwart an attack. *Id.* at [0024]. Hudis teaches that the configuration settings used to thwart the attack can be shared with other deployments. *Id.* at [0024] (“The recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments....***The recommendation system generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.***”); [0025] (“As other deployments within the data center change their configurations to thwart an attack, ***the recommendation system may direct the target deployment to change its configuration accordingly.***”).

244. Hudis also discloses gathering threat intelligence (e.g., IP addresses that should be blocked) from cloud deployments and sharing it with others. *Id.*, [0040] (“A deployment may purchase threat intelligence that, for example, identifies Internet Protocol (“IP”) addresses whose access should be blocked by a firewall.”).

245. These disclosures show that both the system and target cloud deployments receive information describing configuration settings used to thwart an attack. It also would have been obvious to share that information with a cloud [security] management system, for example one operated by the appliance provider, as I discussed earlier.

246. Likewise, Shanks’s cloud management system can “provide users with a management interface that allows users to set configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment.” *See, e.g.*, Ex. 1009 ., [0062]. A POSITA would have readily understood that when someone at a management interface inputs a new configuration using the interface, the cloud management system “receives” it. Accordingly, the “management interface” of Shanks’s cloud management system both transmits and receives configurations, such as firewall settings, for respective CCEs as part of its normal operation.

247. Shanks likewise teaches that cloud management device 502 communicates with CCEs to implement any “configurations, such as security settings, firewall settings, whitelists, blacklists, etc.” *See* Ex. 1008, [0062]. Shanks further teaches “[t]he cloud management system can communicate with computing nodes in client computing environments to implement any received configurations.” Ex. 1009, [0059]. A POSITA would readily have understood that this means the cloud management device receives configurations from customers as a reaction to a security attack, and stores the configurations for use by other customers to combat the security attack. *See* Ex. 1009, [0068]-[0069].

248. Accordingly, the Hudis-Shanks combination teaches 1[c]/9[c]/17[c].

1[d]/9[d]/17[d]: “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment.”

249. The Hudis-Shanks combination teaches “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment,” as recited in 1[d]/9[d]/17[d].

250. Hudis teaches that administrators may change configuration settings for their respective firewalls to thwart an attack: “if several deployments are the subject of a certain type of attack, some administrators of deployments may quickly change the configuration of their firewalls in an attempt to thwart the attack.” Ex. 1008, [0024].

Hudis goes on to teach that “[t]he recommendation system may use such changes in the configurations as a trigger to launch the generating of recommendations for changing the configurations of firewalls of other deployments.” *Id.* The recommendation system then “generates a recommendation relating to the configuration of the firewall for other deployments and provides the recommendation for each deployment.” *Id.* If elected, the system can automatically implement the recommendation configuration. *Id.*; [0025] (“The recommendation system provides a solution to various technical problems encountered when configuring a deployment. For example, a target deployment may use a default configuration for a firewall that is not effective at thwarting certain types of attacks. As other deployments within the data center change their configurations to thwart an attack, ***the recommendation system may direct the target deployment to change its configuration accordingly.***”). More broadly, Hudis teaches that the recommendation system “applies the model to the features of the target deployment to identify a target feature value for the target feature” and “provides the identified target feature value as a recommendation for the target feature for the target deployment.” Ex. 1008, [0004]. This is parallels “identifying . . . one or more configurations to adopt for the first cloud deployment.” And Hudis teaches that the system can “direct a configuration manager of the deployment to change the

configuration to automatically implement the recommended configuration without the explicit approval of an administrator of the deployment.” *Id.*, [0024]. This automatic implementation confirms that the system not only identifies configurations to adopt but can also cause them to be adopted. It would have been obvious to a POSITA that, in order to recommend and, if elected, automatically implement a configuration in a target deployment, the system would have needed to identify a configuration setting to adopt from the configuration settings received from the various cloud deployments. Otherwise, the system could implement unnecessary or irrelevant configuration changes. Additionally, where the deployment is not configured for automatic configuration, an administrator for the deployment would obviously have considered any recommendations and adopted one or more of them. Otherwise, the deployment would be left open to the attack.

251. To the extent this limitation is not taught or obvious over Hudis alone, it would have been obvious over the combination of Hudis and Shanks.

252. Shanks teaches that the cloud management system “automatically configure[s] firewall setting on client computing environments 506 to protect from the security attack.” Ex. 1009, [0068]. For example, Shanks teaches:

To protect from the security attack, cloud management system 502 can apply one or more firewall rules to prevent the security attack. For example, cloud management system 502 can apply a firewall rule to block and/or filter

data packets transmitted from a source associated with the security attack. As another example, cloud management system 502 can apply a firewall rule to block and/or filter data packets transmitted to a destination port associated with the security attack. There [sic] examples are not meant to be limiting. Cloud management system 502 can apply any type of rule to prevent a security attack and this disclosure contemplates any such possibilities.

Ex. 1009, [0069]; *see also id.*, [0060] (“In response to determining that a detected security attack is likely to occur on other client computing environments, the cloud management system can automatically configure firewall setting on client computing environments to protect from the security attack.”). By teaching that the cloud management system applies one or more firewall rules at the CCEs (cloud deployments) to prevent the security attack, Shanks teaches that the cloud management system identifies the one or more firewall rules to prevent the security attack. The one or more firewall rules describe configuration settings used to combat the security threat.

253. As I noted above, a POSITA would have been motivated to combine Hudis and Shanks. In the combination of the two references, this step of identifying could be performed by the recommendation system, cloud [security] management system operated by the appliance provider, or target deployment, depending on the configuration.

254. As such, the Hudis-Shanks combination teaches “identifying, based on the information describing configuration settings used to combat the security threat, one or more configurations to adopt for the first cloud deployment,” as in 1[d]/9[d]/17[d].

7. Claims 2/10/18

2[pre]: “The method of claim 1” / 10[pre]: “The system of claim 9” / 18[pre]: “The computer program product of claim 17”

255. Claim 2 depends from claim 1; claim 10 depends from claim 9; and claim 18 depends from claim 17. The Hudis-Shanks combination teaches claims 1/9/17 as explained above.

2[a]/10[a]/18[a]: “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses.”

256. The Hudis-Shanks combination teaches “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more internet protocol (IP) addresses,” as recited in 2[a]/10[a]/18[a].

257. Shanks teaches that the cloud management system’s management interface “allows users to set configurations, such as security settings, firewall settings, whitelists, blacklists, etc., for their respective client computing environment.” Ex. 1009, [0059]. Shanks further teaches that “[i]n response to determining that a

detected security attack is likely to occur on other client computing environments, cloud management system 502 can automatically configure firewall setting on client computing environments 506 to protect from the security attack.” Ex. 1009, [0068]. Shanks provides various examples of “firewall rules to prevent the security attack” that the cloud management system can apply:

For example, cloud management system 502 can apply a firewall rule to block and/or filter data packets transmitted from a source associated with the security attack. As another example, cloud management system 502 can apply a firewall rule to block and/or filter data packets transmitted to a destination port associated with the security attack. There examples are not meant to be limiting. Cloud management system 502 can apply any type of rule to prevent a security attack and this disclosure contemplates any such possibilities.

Ex. 1009, [0069], emphasis added; *see also id.*, [0074].

258. Additionally, Hudis teaches that deployments can “purchase threat intelligence that, for example, identifies Internet Protocol (“IP”) addresses whose access should be blocked by a firewall.” Ex. 1008, [0040].

259. Given these teachings, including mention of blacklists, firewall rules, and IP addresses that should be blocked, a POSITA would readily have recognized that the configuration setting identifies an IP address to add to a blacklist. Indeed, IP address blacklisting is one of the most fundamental and widely deployed defensive measures in network security, and has been for decades.

260. Accordingly, the Hudis-Shanks combination teaches 2[a]/10[a]/18[a].

8. Claims 3/11/19

**3[pre]: “The method of claim 1” / 11[pre]: “The
system of claim 9” / 19[pre]: “The computer program
product of claim 17”**

261. Claim 3 depends from claim 1; claim 11 depends from claim 9; and claim 19 depends from claim 17. The Hudis-Shanks combination teaches claims 1/9/17 as explained above.

**3[a]/11[a]/19[a]: “wherein the configuration settings
comprise configuration settings that caused the one or
more of the additional cloud deployments to blacklist
one or more network domains.”**

262. The Hudis-Shanks combination teaches “wherein the configuration settings comprise configuration settings that caused the one or more of the additional cloud deployments to blacklist one or more network domains,” as recited in 3[a]/11[a]/19[a].

263. Shanks’s teachings regarding firewall rules are discussed with respect to 2[a]/11[a]/19[a]. *See, e.g.*, Ex. 1009, [0069], [0074].

264. Additionally, as noted with respect to 2[a]/11[a]/19[a], Hudis teaches gathering Indicators of Compromise from deployments and sharing them with others. Before the ’695 patent, well-known indicators of compromise and threat intelligence included suspicious or malicious “network domains.” For example, in 2016, the

NIST published a guide to threat information sharing and identified “a suspicious Domain Name System (DNS) domain name” and “a Uniform Resource Locator (URL) that references malicious content” as examples of indicators. Ex. 1015, 10.

265. Accordingly, a POSITA would have been well aware that the threat intelligence shared with other deployments would include malicious network domains to block. Domain-based blocking was a routine defensive measure well before 2017, and any engineer working in network security would have expected a threat-sharing system to communicate this type of intelligence.

266. Accordingly, the Hudis-Shanks combination teaches 3[a]/11[a]/19[a].

9. Claims 5/13

**5[pre]: “The method of claim 1 further comprising” /
13[pre]: “The system of claim 9, wherein the steps
further comprise”**

267. Claim 5 depends from claim 1 and claim 13 depends from claim 9. The Hudis-Shanks combination teaches claims 1/9 as explained above.

**5[a]/13[a]: “receiving information describing one or
more actions associated with the additional cloud
deployments; and”**

268. The Hudis-Shanks combination teaches “receiving information describing one or more actions associated with the additional cloud deployments,” as recited in 5[a]/13[a].

269. The '695 patent describes an example of an “action”:

In other embodiments, the one or more actions associated with the additional cloud deployments 912 may include one or more user interactions with the additional cloud deployments 912. Such user interactions can include a description of which resources are accessed and/or utilized by particular users, particular business units (e.g., the finance department of a business, the engineering department of a business), particular personas (e.g., a system administrator, a software developer, a human resources manager), and so on.

Ex. 1001, 79:51-59

270. Hudis teaches receiving information describing actions associated with cloud deployments. Hudis teaches that the recommendation system “may monitor the features of the deployments in real time and provide recommendations based on changes in the features of the deployments.” Ex. 1008, [0024]. From such teachings, a POSITA would have understood that the interaction of an administrator to change the configuration of a firewall is such a user action and that the Hudis system continuously monitors for such actions in real time.

271. Hudis further teaches collecting “usage patterns of the resources” (including “patterns of users accessing the deployment, patterns of applications accessing resources of the deployment”) and “logs of the resources and the appliances” (“the logs may indicate when each user logged on to an application of a deployment”). *Id.* at [0040]. These usage patterns and logs describe actions taken by users and applications within the cloud deployments—they constitute “information describing

one or more actions associated with the additional cloud deployments” as recited in 5[a]/13[a].

272. The ’695 patent also describes another example of an “action”:

Likewise, a message, request, or other form of data communications that are exchanged between the additional cloud deployments 912 and actors that are external to the additional cloud deployments 912 may be an example of an action that is associated with the additional cloud deployments 912.

Ex. 1001, 79:41-46.

273. According to Shanks, “security event data” can “include any type of data describing security events detected in client computing devices and/or data that can be used to derive security events that have occurred on a client computing device” and that a “security event can be any type of observable change to the normal behavior of a computing system, device, environment, process, workflow, etc., that can indicate a malicious security breach.” *Id.*, [0063]. A POSITA would have understood or found obvious that an observable changes to a cloud environment that could potentially indicate a security breach would include messages, requests, or other form of data communications with new external third party actors or changes to communication patterns with established external third party actors.

274. For example, any sudden communication activity between any CCE and an external third party would likely be indicative of local malware communicating

outward with a command-and-control node or indicative of an outside party communicating inward attempting to exploit a software vulnerability. If such an observed action was confirmed as a security attack, a likely approach to prevent the same attack in other CCEs would be to configure the firewalls of those other CCEs to disallow any future communication with the suspicious external third party actor.

275. This is precisely the information that Shanks's cloud management device receives from the CCEs. For example, Shanks teaches that CCEs continuously transmit security event data to the cloud management system, which the system analyzes to identify security attacks and prevent them from occurring elsewhere. *See* Ex. 1009, [0060].

276. From such teachings, a POSITA would have understood that Shanks's "security event data" encompasses information describing "actions" associated with client computing devices. A POSITA would have recognized that such "actions" fall within Shanks's teachings of "information describing one or more actions associated with the additional cloud deployments." Put another way, a POSITA would have recognized that in the context of Shanks's cloud management system, the "actions" associated with a cloud deployment are a subset of the security event data the cloud management system is already designed to receive.

277. Accordingly, the Hudis-Shanks combination teaches 5[a]/13[a].

5[b]/13[b]: “wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions.”

278. The Hudis-Shanks combination teaches “wherein identifying the one or more configurations to adopt for the first cloud deployment is further based on the one or more actions,” as recited in 5[b]/13[b].

279. As discussed for 1[d]/9[d]/17[d], the Hudis-Shanks combination teaches “identifying...one or more configurations to adopt for the first cloud deployment.”

And, as discussed for 5[a]/13[a], the Hudis-Shanks combination teaches that the received security event data describes one or more actions associated with the additional cloud deployments.

280. Shanks teaches using the received security event data, which is continuously transmitted from CCEs, to determine that a detected attack is likely to occur elsewhere and, in response, selecting protections and configuring firewall settings for the CCEs that have opted in:

In addition to enabling users to easily manage their client computing environments, the cloud management system can also analyze aggregated data to automatically configure firewalls based on aggregated cloud managed information. For example, client computing environments can continuously transmit security event data to the cloud management system. **The cloud management system can analyze the security event data to identify security attacks and determine whether the security attacks are likely to occur on other client computing environments.**

In response to determining that a detected security attack is likely to occur on other client computing environments, the cloud management system can automatically configure firewall setting on client computing environments to protect from the security attack.

Ex. 1009, [0060].

281. Because Shanks teaches using the security event data to identify configurations to adopt for the first cloud deployment as per 1[d]/9[d]/17[d], and because Shanks teaches that the security event data includes one or more actions as per 5[a]/13[a], the Hudis-Shanks combination teaches 5[b]/13[b]. Additionally, as discussed for 5[a]/13[a], Hudis collects administrators changing firewall configurations. Naturally, the configuration to adopt in another cloud deployment would be based on that.

7. Claims 8/16

**8[pre]: “The method of claim 1 further comprising” /
16[pre]: “The system of claim 9 wherein the steps
further comprise”**

282. Claim 8 depends from claim 1; and claim 16 depends from claim 9. The Hudis-Shanks combination teaches claims 1/9.

**8[a]/16[a]: “receiving information describing one or
more deployment processes associated with the
additional cloud deployments.”**

283. The Hudis-Shanks combination teaches “receiving information describing one or more deployment processes associated with the additional cloud deployments,” as recited in 8[a]/16[a].

284. Fortinet’s Infringement Contentions allege:

Netskope Cloud Exchange may receive information from third party cloud deployments that can include information concerning the name of the file associated with an identified threat, which is insubstantially different from information concerning the deployment processes employed in the cloud deployment as the file name (as incorporated in the “comment” field) can establish the component (e.g., program) within the cloud deployment associated with the identified threat.

Ex. 1016, 43. Fortinet evidently considers receiving “information concerning the name of the file associated with an identified threat”—that is, a file name—to be essentially the same as receiving “information concerning the deployment processes employed in the cloud deployment.”

285. At the time of the alleged invention, the technique of scanning incoming files to avoid importing malware to a computing device had been in use for approximately 30 years. Some of the earliest PC anti-virus utilities matched known infected program names and sizes as a crude heuristic, since the first DOS viruses often targeted specific files (e.g., COMMAND.COM) or appended to.COM/.EXE files. A POSITA would have understood that file names, especially file extensions, were

useful for identifying potential threats, but that this was a very primitive technique that was long augmented by analyzing files' actual contents, e.g., by comparing hashes of file contents with hashes of known threats.

286. Although Shanks does not explicitly disclose using file names as part of “security event data,” a POSITA would have well understood that file names, extensions, and hashes were valuable and usable information related to security threats and that their uses in detecting and mitigating such security threats were conventional and had been in widespread use for decades. (Ex. 1015 at 10). To a POSITA, it would have been totally obvious to include file names, file extensions, and file content hashes among the set of “security event data” that one would use to detect and counteract a threat.

287. Accordingly, the Hudis-Shanks combination teaches “receiving information describing one or more deployment processes associated with the additional cloud deployments,” as recited in 8[a]/16[a].

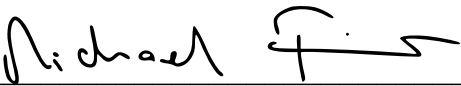
IX. CONCLUSION

288. For the reasons set forth above, it is my opinion that claims 1-3, 5, 8-11, 13, 16-19 of the '695 patent would have been obvious to a POSITA at the time of the alleged invention in view of the above grounds.

Declaration of M. Franz ISO Petition for
Inter Partes Review Of U.S. Pat. No. 12,126,695

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; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

This declaration was executed in Berlin, Germany on September 30, 2026.



Michael Franz