IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

TREND MICRO, INC.
Petitioner,

v.

TAASERA LICENSING LLC
Patent Owner.

_____

IPR2023-00800

U.S. Patent No. 8,990,948

DECLARATION OF SETH JAMES NIELSON

TABLE OF CONTENTS

I. INTRODUCTION 1

II. QUALIFICATIONS 1

III. MATERIALS CONSIDERED 12

IV. LEGAL STANDARDS 13

V. Technical Background that Would Have Been Known to A POSITA 16

A. Intrusion Detection Systems 16

B. Vulnerability Scans and Assessments.....17

C. Applications, services, and DLLs 18

VI. OVERVIEW OF THE CHALLENGED PATENT 20

A. THE ’948 PATENT SPECIFICATION AND CLAIMS 20

B. PROSECUTION HISTORY 25

C. CLAIM CONSTRUCTION .....**Error! Bookmark not defined.**

1. Native computing environment (Claim 1) ..... **Error! Bookmark not defined.**

VII. LEVEL OF SKILL IN THE ART 27

VIII. Overview of the Prior art 27

A. OVERVIEW OF “NETWORK INTRUSION DETECTION” BY MUKHERJEE (“MUKHERJEE”) 27

B. OVERVIEW OF US 7,451,488 TO COOPER (“COOPER”) 31

IX. GROUND 1: Claims 1-5 are rendered obvious by MUKHERJEE IN VIEW OF COOPER 33

A. Motivation to Combine Mukherjee and Cooper 33

B. Claim 1 is rendered obvious by Mukherjee in view of Cooper 38

1. Limitation 1[a] 38

2. Limitation 1[b] 47

3. Limitation 1[c] 51

4. Limitation 1[d] 61

5. Limitation 1[e] 64

C. Claim 2 is rendered obvious by Mukherjee in view of Cooper 73

D. Claim 3 is rendered obvious by Mukherjee in view of Cooper 74

E. Claim 4 is rendered obvious by Mukherjee in view of Cooper 77

F. Claim 5 is rendered obvious by Mukherjee in view of Cooper 81

X. There are no objective indicia of nonobviousness 87

XI. Conclusion 87

1. I, Seth James Nielson, submit the following declaration (the "Declaration") in connection with the proceeding identified above.

I. INTRODUCTION

2. I have been retained by counsel for Trend Micro, Inc. ("Trend Micro" or "Petitioner") as a technical expert in connection with the proceeding identified above. I submit this Declaration in support of Petitioner Trend Micro's Petition for Inter Partes Review (the "Petition") of United States Patent No. 8,990,948 ("the '948 patent") against Taasera Licensing LLC ("Taasera"). All "Ex. 10XX" cites herein are to the Exhibits to the Petitions. All citations to "Section XX" are internal citations to the sections of this Declaration. All citations to "Paragraph XX" are internal citations to the paragraphs of this Declaration.

3. I have been asked to consider whether the claims of the '948 patent are obvious in light of the prior art discussed below. The '948 patent is directed to general application of known computer security analysis techniques to the detection of threats in a network.

4. In summary, my conclusion is that the claims of the '948 patent are obvious in light of the prior art discussed below. As set forth in detail in this Declaration, I have reached the following conclusions:

a. Claims 1-5 are rendered obvious by the combination of "Network Intrusion Detection" by Mukherjee et al. ("Mukherjee") in view of U.S. Patent No. 7,451,488 to Cooper et al. ("Cooper").

II. QUALIFICATIONS

5. I am a subject matter expert in cybersecurity, computer networks, and software engineering. I am the Founder and Chief Scientist of Crimson Vista, a computer security research and engineering company. Furthermore, I hold appointments at the University of Texas at Austin

as an Adjunct Associate Professor in the department of Computer Science and as a Cybersecurity Fellow in the Robert Strauss Center for International Security and Law.

6. I have been working with computer networks and associated technology, including parallel processing, since January 2000. My experience includes graduate-level teaching, academic research, industry employment, and consulting practice.

7. Exhibit 1004 is a current copy of my C.V. and includes all of my relevant education and work experience, a list of all publications that I have authored in the previous ten years, as well as a list of all cases in which, during the previous four years, I have testified as an expert at trial or by deposition. My experience most relevant to the opinions offered in this report is summarized below.

8. I received my B.S. in Computer Science in April of 2000. During my final undergraduate semester, I worked both as a teaching assistant for a Computer Networking course and as a researcher in the Networked Computing Lab. In these capacities, I assisted students in debugging and designing TCP/IP protocol stacks, Address Resolution Protocol implementations, and Remote Procedure Call projects. In the research lab, I collaborated on work related to resource reservations in high performance networking and MPLS analyses. Our investigations into statistical traffic engineering for bandwidth allocation resulted in a published paper entitled, "Effective Bandwidth for Traffic Engineering."

9. During the Fall of 2000, I studied for one semester at Brandeis University in the Boston area and, as part of my research, I developed a parallel-processing system for DNA string matching. The system that I created had a central node that managed the search space and solution aggregation. A cluster of available "worker" machines registered with the central node to indicate their availability for parallel processing. The central node would identify subdivisions of the search

space that could be farmed out to the worker nodes for processing. I also studied artificial intelligence and machine learning.

10. From 2001 through 2003, I worked as a software engineer at Metrowerks (formerly Lineo, Inc.), where I had substantial responsibilities relating to software architecture, computer networking, and technical project management. In particular, I developed and maintained the GUI for the Embedix SDK (Software Development Kit), created a network communication system between a Windows front-end and Linux back-end, developed an automated system to forward Linux python scripts to a Windows GUI over the network, and developed a network-deployed packaging and automated updating system for client software. To complete these assignments, I wrote tens of thousands of lines of computer code in C++, C, Python, and Perl.

11. As part of my technical training and development at Lineo/Metrowerks, I also gained and employed an increased exposure to computer networking and network security. I tested and evaluated a prototype firewall product, built a custom VPN solution, and trained in the use of web server and mail server administration (including security). I ran my own personal MTA with an IMAP and POP3 server based on these tools under development. I also deployed a Squid web proxy for both caching and content filtering.

12. Another major networking project involved porting the Embedix SDK from Linux to Windows. For this project I used a virtual-machine/networked solution. The underlying SDK engine remained running in a Linux VM while the GUI operated in native Windows. I developed all of the code for this remote communication system.

13. While working at Lineo/Metrowerks, I also returned to BYU to pursue my Master's degree in Computer Science. In addition to the graduate level course work in wireless computer networks and compilers, I pursued graduate research in software engineering topics, with a special

emphasis on how programmers think while creating and modifying code. During my course work, I took a special topics class called "Programmer Cognition" as well as a graduate-level neuroscience class from the Psychology department.

14. My research included a study of computer architectural patterns and how those patterns might need to change as programming languages change and evolve. Based on my research, I proposed a concept called "Design Dysphasia," wherein a programmer or software developer becomes trapped in their approach to solving problems based on the paradigms and design approaches of the programming language. My research was published as "Design dysphasia and the pattern maintenance cycle," in the Journal Information and Software Technology August 2006. This work also was a major component of my Master's thesis.

15. Another part of my Master's thesis was the identification of how certain programming language concepts can be "mixed" together. I investigated practical mechanisms whereby the Python programming language could be extended to support features known as "functional programming" and "logic (or declarative) programming." Languages with this mix of features are known as "multi-paradigm" programming languages.

16. After finishing my Master's degree, I moved to Houston TX in 2004 to begin a PhD program at Rice University. I chose to focus my research into topics of network security including the security of peer-to-peer cooperative storage and computation technologies.

17. During the 2004 fall semester of my Ph.D. program at Rice University, I identified a security vulnerability in the Google Desktop Search that could have allowed hackers to compromise users' computers and obtain private information. After contacting Google and assisting them in closing the vulnerability, we published the details of our investigation.

18. In 2005, I completed an internship at Google, where I designed and implemented a solution to privacy loss in Google Web Accelerator. The Google Web Accelerator was designed to increase the speed of browsing the Internet. Once installed on a user's computer, the browser would request all content through a Google Proxy. The proxy performed pre-fetching and extensive caching in order to provide fast and responsive service to the user. At the time of my internship, news reports had identified odd problems in which users of the Accelerator were accessing other individual's private pages. During my internship, I designed and implemented a prototype solution for this issue in C++.

19. In 2005, I published a paper entitled, "A Taxonomy of Rational Attacks." This paper categorized and described the various types of attacks that one might see in a decentralized, peer-to-peer network. When there is no centralized authority, users have to cooperate to obtain service. The term "rational attacks" refers to the economic incentives to not cooperate while still exploiting the system for service.

20. My Ph.D. Thesis was entitled "Designing Incentives for Peer-to-Peer Systems," and it built on this concept. Given a network where participants cannot be forced to cooperate, the operation of said network must induce cooperation by design of the outcomes. In other words, it must be in each participant's best interest to contribute to the cooperative operation. I conducted experiments including simulated extensions to the BitTorrent peer-to-peer protocol for long-term identities and mechanisms for cooperative anonymity. These systems were predecessors to modern cryptocurrency and distributed ledger technologies.

21. From 2005 through 2008, with the approval of my PhD adviser, I worked as a Security Analyst for Independent Security Evaluators (ISE). Much of my early work was spent developing a software encryption library, including the necessary tests and procedures for FIPS-

certification. The encryption library provided advanced operations such as secure data splitting and recovery.

22. In 2009, I went to work full time for ISE as a Security Analyst and later as a Senior Security Analyst. I built a number of advanced projects including a parallel-program for massive code coverage analysis, GPU hardware-accelerated AES encryption, and distributed file-system prototypes.

23. In addition to the software development, I also performed security evaluation services that included port-scanning analyses, security protocol analysis using formal and exploratory methods, and investigated security breaches.

24. I also designed and managed the implementation of a secure communication technology that splits trust between multiple SSL Certificate Authorities (CA), so that if one CA is compromised, the communication stream can still be safely authenticated. My work on the secure communications technology project led to the issuance of multiple patents. In total, I wrote hundreds of thousands of lines of code in C, C++, and Python, including projects where I had to implement the same functionality in two separate languages.

25. In 2011, I began work as a Research Scientist at Harbor Labs and continued with that consulting firm until fall 2015. I worked with a wide range of clients, specializing in network security, network communications, software architecture, and programming languages. I analyzed an extensive collection of commercial software, including software related to secure email, cloud-based multimedia delivery, document signing, anti-virus and anti-intrusion, high-performance routing, networking protocol stacks in mobile devices, PBX telecommunications software, VoIP, and peer-to-peer communications. I also analyzed security considerations for potential technology acquisitions.

26. Also at Harbor Labs, I reviewed technology and source code for multiple clients related to accusations of theft and/or misappropriation of trade secrets. These engagements included an analysis of C, C++, Java, Python, and other source code languages in high-frequency trading, e-commerce, and other similar systems.

27. I also assessed the security and privacy technologies and policies provided by a third-party vendor to the Center for Copyright Infringement (CCI). CCI represents content owners, such as the RIAA and the MPAA, in finding and reducing piracy online. Because this process necessarily involves collecting information about private individuals by scanning a network for illegal activity, I was asked to investigate and determine that the information collected from online computing devices was adequately safeguarded and protected.

28. For other clients, I have "resurrected" or re-created legacy software systems. For example, I helped one client make remote desktop-sharing/viewing systems from the mid 90's operational. I helped them identify the most compatible components from an old CVS repository, obtain the necessary legacy hardware and software to rebuild the source code, and diagnose why the separate components weren't completely compatible with each other. Using tools from the era (i.e., the mid-90's), I identified and fixed these issues in C++ and Java code, and successfully demonstrated the operational system across a small cluster of networked virtual machines.

29. During my final year at Harbor Labs, I was engaged as the principal consultant with a large biomedical device firm in a twelve-month analysis of the security of their products. Notably, medical devices were for some time not considered significant threats in terms of computer security. However, recent demonstrations by security researchers of the various ways in which a malicious individual might harm a person using a medical device has shifted the thinking in the industry. Accordingly, I was engaged to assist this company in the analysis of their products,

their process, and their future roadmap in order to ensure that patients are not harmed. I and my team analyzed design documents, hardware, and a broad range of additional resources in order to expose potential problems. The security of these systems depends, in part, on the architecture and deployment of the networks in which they operate.

30. In December 2015, I left Harbor Labs to assist Ironwood Experts, LLC., as the transitional managing partner. In three months, I helped to establish a new direction, streamline operations, and wrap up difficult negotiations.

31. After handing off management responsibilities at Ironwood, I founded Crimson Vista, Inc. as a boutique computer security engineering company. Similar to the work that I did at Harbor Labs, I continue to provide technical expertise to a wide range of clients in areas of programming languages, computer networks, and network security. My expertise in the area of "security engineering" provides comprehensive analysis, design, and insight into cybersecurity concerns before, during, and after development.

32. For example, I have been retained by a start-up in telecommunications security to provide cryptography expertise and evaluations of their protocols and architectures. My team and I have prototyped new protocols, written up analyses, and presented to potential partners and investors.

33. Another start-up company retained me for guidance in matters relating to Blockchain and Smart Contracts. This technology is very much dealing with a "fad" phase where there is a lot of misinformation and hype. I guided the start-up company through analyzing where these kinds of technologies would help and where they would not. I have also provided training on Blockchain at the Data Architecture Summit and Enterprise Data World conferences.

34. I have also provided technical guidance to an antitrust team in the United States Department of Justice. Although the technologies and parties are confidential, I can disclose that I provided in-person training on technical topics and analyses of competing security products.

35. More recently, I been retained by clients, including a Fortune 100 financial institution, to provide them with post-data-breach analyses of what went wrong, the impact of the lost data, and guidance on resolution. In these engagements, I provided reverse engineering of the data to demonstrate how an attacker can or would use the compromised information, analyzed software development to determine when the system became vulnerable, and helped identify impacted customers that had been missed in the investigations.

36. I have also been retained as a vCISO (virtual Chief Information Security Officer) for a small company that needs security expertise and guidance in protecting their newly developed intellectual property. I advise the company on security policy, operations, implementations, and training.

37. Through Crimson Vista, I also invest in research and development. Recent projects include engaging with a partner to implement prototypes of communications security protocols for next-generation automobiles. I also gave a talk on "Detecting Malicious Sandboxes" at the Workshop on Defensive Deception and Trust in Autonomy, in association with the 2018 Naval Applications of Machine Learning Workshop.

38. I was the primary investigator on a government funded research project. The project dealt with automatic recovery from ransomware attacks and was funded by the United States Army. I led a team of student researchers from Johns Hopkins and Brigham Young University in developing my design and implementing a prototype.

39. I am also a contributor to another small-business research project begun by a former student and funded by the National Science Foundation. We are researching the development of a new system for secure data storage that is protected even against "insider" threats, such as a rogue IT administrator. As part of this work, I am managing several interns for Summer 2022 and directing their work on performance, analysis, and vulnerability detection.

40. I also continue to perform a wide range of code reviews for diverse technologies including CAD software, video game systems, digital mobile radios (DMRs), video streaming, and digital rights management (DRM). I am often retained for my expertise in software forensics and have been instrumental in multiple cases to discovering and identifying stolen source code or misappropriated trade secrets.

41. Moreover, I maintain ties to academia. I held adjunct appointments at Johns Hopkins University from 2014 to 2019. From July 2016 to July 2019, I also held an appointment as the Director of Advanced Research Projects in the Johns Hopkins University Information Security Institute.

42. At Johns Hopkins University I taught Network security and Advanced Network Security. I created a custom curriculum and lab experience wherein students developed their own security protocols as a class and then attempted to break their own creations. Students learned how hard it is to get security right, and how easy it is to find something wrong. I published a paper about the labwork in the Journal of Computer Science Education entitled, "PLAYGROUND: Preparing Students for the Cyber Battleground."

43. Beyond course instruction, I also mentored Masters students at Johns Hopkins in their capstone projects. These projects included networking security and privacy concerns across a wide range of technologies including cryptography, drone security, iOS security, BitCoin, SSL

vulnerabilities, and Twitter "botnets." My students and I have published multiple papers from these capstone projects.

44. During my tenure as the Director of Advanced Research Projects, I was tasked with developing collaborative research opportunities. Through my efforts, a wide range of student capstones have been executed with partners from the Johns Hopkins Applied Physics Lab or outside corporate partners.

45. For example, we coordinated with the company OnBoard Security to develop better security for anti-collision protocols for air traffic. Students demonstrated the potential issues related to leaving the protocol unsecured and built a working prototype of a secured variant. OnBoard and Johns Hopkins published press releases which were picked up by aviation-focused news sources.

46. I am now an adjunct professor at the University of Texas at Austin. I have taught the undergraduate Network Security and Privacy class in the Computer Science department. I also teach the Introduction to Cybersecurity Technology class in the Law School

47. I am also the co-founder and current director of the Crypto Done Right project. This project is currently hosted by Johns Hopkins University and funded by a grant from Cisco. I am transitioning the project to be hosted by my company Crimson Vista or spun out into a completely independent non-profit entity. Crypto Done Right is designed to bridge the gap between cryptography SME's and the IT professionals that use it. It provides authoritative guidance on deployment, lifecycle, and management of cryptography in IT systems, software development, and technical management.

48. Finally, I am the author of "Cryptography in Python: Learning Correct Cryptography by Example." I am authoring a second book on computer security that will be published later this year.

III. MATERIALS CONSIDERED

49. I have considered information from various sources in forming my opinions. Besides drawing from approximately two decades of experience in the computer industry, I also have reviewed the '948 patent and its file history, the other documents and references as cited herein, and general technical references.

50. I understand that Counsel for Petitioner have submitted a Petition that references by exhibit number. For simplicity and convenience, I have adopted the same exhibit numbering. The exhibits are:

- Ex. 1001: U.S. Patent No. 8,990,948 to Kumar et al. ("the '948 patent")
- Ex. 1002: Prosecution history of the '948 patent
- Ex. 1003: Curriculum Vitae of Seth James Nielson, Ph.D.
- Ex. 1004: Expert Declaration of Seth James Nielson, Ph.D.
- Ex. 1005: Biswanath Mukherjee, Network Intrusion Detection, IEEE Network, (June 1994). ("Mukherjee")
- Ex. 1006: U.S. Patent No. 7,451,488B2 to Cooper et al. ("Cooper")
- Ex. 1007: William R. Cheswick, Firewalls and Internet Security Second Edition, Repelling the Wily Hacker 2, (2003).
- Ex. 1008: Ross Anderson, Network Attack and Defense, Security Engineering: A Guide to Building Dependable Distributed Systems 2, 633-678, (2008).

- Ex. 1009: Andrew S. Tanenbaum, Modern Operating Systems 3, Pearson Prentice Hall, (2008).
- Ex. 1010: U.S. Patent Publication No. 2008/0175266 to Alperovitch et al.
- Ex. 1011: U.S. Patent Publication No. 2011/0145711 to Njemanze et al.
- Ex. 1012: U.S. Patent No. 5,604,862 to Midgely et al.

IV. LEGAL STANDARDS

51. I am not an attorney, and I have relied on instructions from counsel as to the applicable legal standards to use in arriving at my opinions in this Declaration.

52. I have been informed and understand that patent claims are construed from the perspective of one of ordinary skill in the art at the time the claimed invention was made and that, during this proceeding, claims are construed using the same claim construction standard that would be used to construe the claim in a civil action. Under that standard, claim terms are to be given their plain and ordinary meaning, which is the meaning understood by a POSITA in light of the claim language, patent specification, and prosecution history.

53. I understand that a patent may include independent claims and dependent claims. An independent claim stands by itself and only includes the limitations it recites. A dependent claim can depend on (a) an independent claim, or (2) another dependent claim. I understand that a dependent claim includes all of the limitations it recites plus the limitations recited in the claim from which it depends.

54. I have been informed and understand that the subject matter of a patent claim is obvious if the differences between the subject matter of the claim and the prior art are such that the subject matter of the claim as a whole would have been obvious at the effective filing date to a person having ordinary skill in the art to which the subject matter pertains. I have also been

informed that the framework for determining obviousness involves considering the following factors: (i) the scope and content of the prior art; (ii) the differences between the prior art and the claimed subject matter; (iii) the level of ordinary skill in the art; and (iv) any objective evidence of non-obviousness.

55. I further understand that the claimed subject matter would have been obvious to one of ordinary skill in the art if, for example, it results from the combination of known elements according to known methods to yield predictable results, the simple substitution of one known element for another to obtain predictable results, the use of a known technique to improve similar devices in the same way, the application of a known technique to a known device ready for improvement to yield predictable results, or choosing from a finite number of identified, predictable solutions with a reasonable expectation of success. I have also been informed that the analysis of obviousness may include recourse to logic, judgment, and common sense available to the person of ordinary skill in the art that does not necessarily require explication in any reference. I understand that so-called "secondary considerations of non-obviousness," must be considered in an obviousness analysis. I understand that an analysis cannot use hindsight in determining whether a patent claim is obvious. I understand that secondary considerations of non-obviousness include: (a) a long-felt but unresolved need for the invention; (b) commercial success of the invention; (c) copying of the invention; (d) praise and recognition of the invention by others; (e) licensing of the rights to the invention; and (f) unexpected results.

56. I understand that obviousness must be determined with respect to the challenged claim as a whole.

57. I understand that a challenged claim can be invalid for obviousness over a combination of prior art references if a reason existed (at the effective filing date of the Patent in

question) that would have prompted a person of ordinary skill in the art to combine prior art elements in the way required by the claim in question. I understand that this requirement is sometimes called "motivation to combine." I further understand that the prior art references themselves might provide a motivation to combine, and that sometimes commonsense as would have been known to a POSITA can provide a reason to combine the disclosures of two or more prior art references. I also understand that market demand, rather than scientific literature, often drives innovation, so that market forces can give rise to a motivation to combine. I also understand that known work in a field might prompt variations of it to use in the same field or a different one based on design incentives or market forces if the variations would have been predictable to a POSITA. I also understand that a need or problem known in the field can provide a reason for combining elements in the manner claimed.

58. I understand that in order for a claim to be found invalid based upon a modification or combination of prior art references, a POSITA must have had a reasonable expectation of success in modifying or combining the prior art.

59. I further understand that claims are invalid as obvious if they merely recite combinations of, or obvious variations of, prior art teachings employing the understanding and knowledge of a POSITA and motivated by the general problem that the inventor faced at the time. The prior art references themselves, or any need or problem known in the field at the time of the invention can provide a motivation to combine the elements from prior art references or to attempt obvious variations on them in the claimed manner.

60. I understand that something is "inherent in" the prior art—and thus taught by it—it if necessarily follows from the prior art's explicit disclosures.

61. I understand that the filing date of a U.S. patent is generally the filing date of the U.S. application that issued as a patent. I also understand, though, that a U.S. patent may be granted an earlier effective filing date if the patent owner properly claimed priority to an earlier patent application.

V. TECHNICAL BACKGROUND THAT WOULD HAVE BEEN KNOWN TO A POSITA

62. In this section of my Declaration, I discuss some of the background and context for the alleged invention in the '948 patent that would have been known to a POSITA.

A. Intrusion Detection Systems

63. Computer networks frequently contain confidential data or grant access to sensitive operations that the operators of such networks would like to protect and that attackers would like to exploit. It is not always possible or practical to prevent such attacks before they occur, so one tool to detect ongoing attacks is called an Intrusion Detection System (IDS). "The systems used to detect bad things happening are referred to generically as intrusion detection systems… the term is most usually applied to boxes that sit on your network and look for signs of an attack in progress or a compromised machine [1100]." (Ex. 1008 at p. 33).

64. "There are several types of intrusion detection systems. Network IDSs (NIDSs) eavesdrop on network traffic, looking for an indication of an intrusion. Various host-based systems scan files or traffic for incoming viruses; some analyze system call patterns or scan for changed files." (Ex. 1007 at p. 9). Still others analyze audit logs of a host's operating system to detect potentially malicious behavior.

65. Furthermore, the various types of IDSs are often employed in concert: "As noted, IDSs can be host-based or network-based. The two are complementary, not mutually exclusive; each has its strengths and weaknesses. Host-based systems tend to know the state of their own

machine, which simplifies the processing of the data flows, but the software can be subverted if the host is compromised. Network-based devices are stand-alone units and are presumably more resistant to attack or even detection." (*Id.* at p. 11). In addition, various types of IDSs operate using different techniques: "Some products try to apply AI techniques to the problem. Others collect distributed information and try to assemble an overall view of an attack." (*Id.* at p. 12).

B. Vulnerability Scans and Assessments

66. Software running on a network-connected computer may have bugs that expose exploitable vulnerabilities to attackers. Public databases of such known vulnerabilities exist, such as Common Vulnerabilities and Exposures (CVE). In order to discover whether a computer is affected by any of these known vulnerabilities, attackers and system administrators alike may use scanning tools. (*See* Ex. 1007 at p. 3).

67. The goal of network scanners is to discover hosts on the network and learn about the capabilities and vulnerabilities of those hosts. Once such tool is called *nmap*. "It has an extensive database of TCP/IP stack idiosyncrasies for many versions of various operating systems… The database can be quite useful on its own—companies are quite interested in inventory and version control, and nmap has the best database we know for host fingerprinting, or identifying the operating system and version from afar." (*Id.* at p. 7). The purpose of such fingerprinting is to determine the version of software running on the machine. If, for example, a machine is running an old version of a piece of software (e.g., an operating system) with a known vulnerability, an attacker can recognize that the host is vulnerable to that particular attack. Likewise, a system administrator can recognize that the host lacks an up-to-date patch. (*See id.* at pp. 5-6).

68. Another technique for vulnerability scanning is port scanning. "Hosts are also scanned for active ports. They seek active network services, and often identify the server software

and versions." (*Id.* at p. 6). Various tools have been developed for this technique over the years. "The first port scanner we are aware of was a set of shell scripts written by Mike Muus around 1988. ISS followed in the early 1990s, and then SATAN. Now Nessus is available from http://www.nessus.org." (*Id.* at p. 8).

C. Applications, services, and DLLs

69. Applications developed for major consumer operating systems rely heavily on shared libraries, Application Programming Interfaces (APIs), and system calls to do their work. For example, with respect to system calls, "Microsoft has defined a set of procedures called the Win32 API (Application Program Interface) that programmers are expected to use to get operating system services." (Ex. 1009 at p. 4). These system calls involve things like interacting with the filesystem, creating processes, and retrieving the clock time. (*Id.* at p. 5). Unix and Linux systems have similar techniques for exposing such system calls. (*Id.* at pp. 3-5).

70. Developers also make use of these APIs for managing user interfaces. For example, on Windows, "[t]he Win32 API has a huge number of calls for managing windows, geometric figures, text, fonts, scrollbars, dialog boxes, menus, and other features of the GUI [Graphical User Interface]." (*Id.* at p. 4). These functions are often encapsulated in libraries called Dynamic Link Libraries (DLLs), whose key feature is the ability to share functionality between programs. Making these libraries available to multiple programs enables code reuse, consistency, and efficiency: "DLLs improve the efficiency of the system by allowing common code to be shared among processes, reduce program load times from disk by keeping commonly used code around in memory, and increase the serviceability of the system by allowing operating system library code to be updated without having to recompile or relink all application programs that use it." (*Id.* at p. 7). Such libraries may be written by the operating system manufacturer or by third-party developers, and they provide shared code available to many programs.

71. As another example, programs may allow themselves to be used as building blocks for other programs: "Applications like Microsoft Office link and run DLLs to allow Office to be used as a platform for building other applications. The COM (component object model) style of programming allows programs to dynamically find and load code written to provide a particular published interface, which leads to in-process hosting of DLLs by almost all the applications that use COM." (*Id.* at p. 8).

72. Sometimes such shared libraries may even run as services (i.e., processes that offer shared functionality to many programs on the computer). For example, "Windows Vista makes significant use of user-mode service processes to extend the functionality of the system. Some of these services are strongly tied to the operation of kernel-mode components, such as lsass.exe which is the local security authentication service which manages the token objects that represent user-identity, as well as managing encryption keys used by the file system. The user-mode plug-and-play manager is responsible for determining the correct driver to use when a new hardware device is encountered, installing it, and telling the kernel to load it. Many facilities provided by third parties, such as antivirus and digital rights management, are implemented as a combination of kernel-mode drivers and user-mode services." (*Id.* at p. 9). Some of these services may themselves be DLLs: "Within each of the shared service processes, individual services are loaded as DLLs." (*Id.*).

73. DLLs and services are also particular targets for malicious activity, making them salient in analyses that look for such anomalies. For example, with respect to DLLs:

> On the other hand, shared libraries introduce the problem of versioning and increase the complexity of the system because changes introduced into a shared library to help on particular program have the potential of exposing latent bugs in other applications, or just breaking them due to changes in the implementation—a problem that in the Windows world is referred to as DLL hell.

(*Id.* at p. 7).

74. Furthermore, with respect to services:

> Services are common sources of security vulnerabilities in the system because they are often accessible remotely (depending on the TCP/IP firewall and IP Security settings), and not all programmers who write services are as careful as they should be to validate the parameters and buffers that are passed in via RPC. The number of services running constantly in Windows is staggering. Yet few of those services ever receive a single request, though if they do it is likely to be from an attacker attempting to exploit a vulnerability.
> (*Id.* at p. 9).

75. Thus, applications run in many forms, whether a large, well-known end-user software package like Microsoft Office, a service that manages device drivers, or a DLL containing shared functions or system calls.

VI. OVERVIEW OF THE CHALLENGED PATENT

A. THE '948 PATENT SPECIFICATION AND CLAIMS

76. U.S. Patent No. 8,990,948 ("the '948 patent") to Srinivas Kumar and Dennis Pollutro was filed on July 27, 2012 and issued on March 24, 2015. The patent, titled "Systems and Methods for Orchestrating Runtime Operational Integrity," relates to the application of computer security analysis techniques to the detection and remediation of threats and risks in a network. (Ex. 1001 at Abstract). In particular, it discloses methods of monitoring, correlating, and displaying network security information. (*See* Ex. 1001 at 4:28-44).

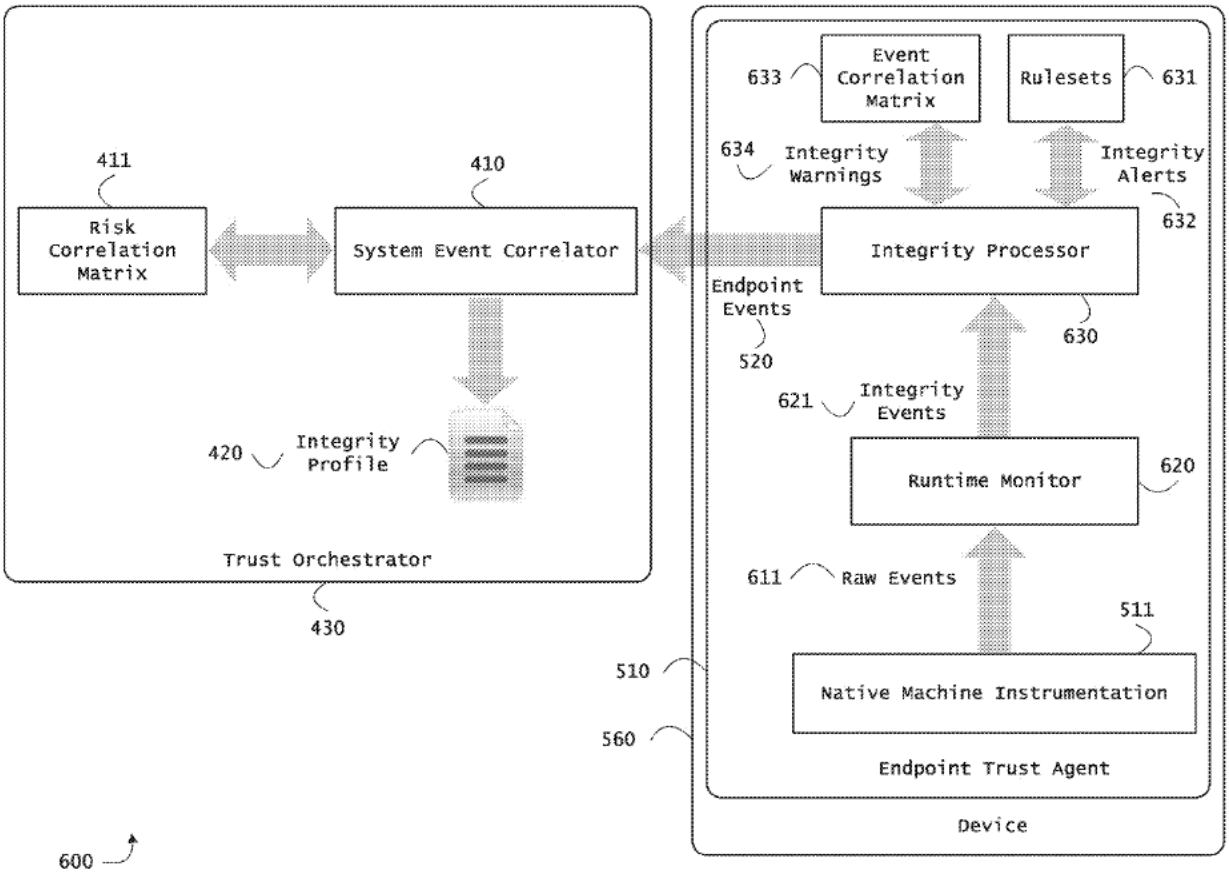
77. The '948 patent recognizes that "[t]he proliferation of applications (business, social networking and gaming software) in Enterprise and cloud computing ecosystems has significantly increased the attack surface and window of exposure," and "[t]he proliferation of unmanaged devices such as mobile computing devices (i.e., tablets and smartphones) and Bring Your Own Device (BYOD) policies in Enterprise network environments has increased the risk of advanced coordinated threats." (*Id*. at 2:15-24). As such, the patent claims disclose methods with the goal

of "providing real-time operational integrity of an application on a native computing environment". (*Id.* at Claim 1).

78. The method accomplishes this by "monitoring, by a plurality of sensory inputs" data about network activity, system operations, application runtime configuration, resource utilization, and application integrity. (*Id.* at Claim 1). The method continues by generating events to ascertain "real-time operational integrity of the application" through inputs from "a network analyzer, an integrity processor, an event correlation matrix, and a trust supervisor." (*Id.*). Subsequently, the method correlates the events and displays, to an administrator, information about the application operational integrity. (*Id.*). The '948 patent claims and specifications also disclose various embodiments which specify the types of events and other data to be processed, the frequency with which such processing occurs, and the manner in which the data is displayed. (*See id.* at Claims 2-5).

79. An exemplary system or environment for the alleged invention, shown in Figure 6, includes a device, which "may be a desktop, server, laptop, a mobile computing device such as a smartphone or tablet, or other types of mobile computing platforms that comprise (a) an operating system (OS); (b) a user operating the device to access a service or another device 560 over a network; and (c) application packages, applications or application components that may be downloaded and installed either with or without user intervention for later execution." (*Id.* at 13:16-25). The device 560 includes "native machine instrumentation 511 [which] can represent any event subscription, callback, notification mechanism provided by the supported operating system (OS) on the device 560 (e.g, MICROSOFT™ WINDOWS® Machine Instrumentation (WMI), Transport Filter Drivers, Application Layer Enforcement Callout Drivers, Linux/Unix Network Filter Drivers, etc.)." (*Id*. at 13:43-50).

FIG. 6

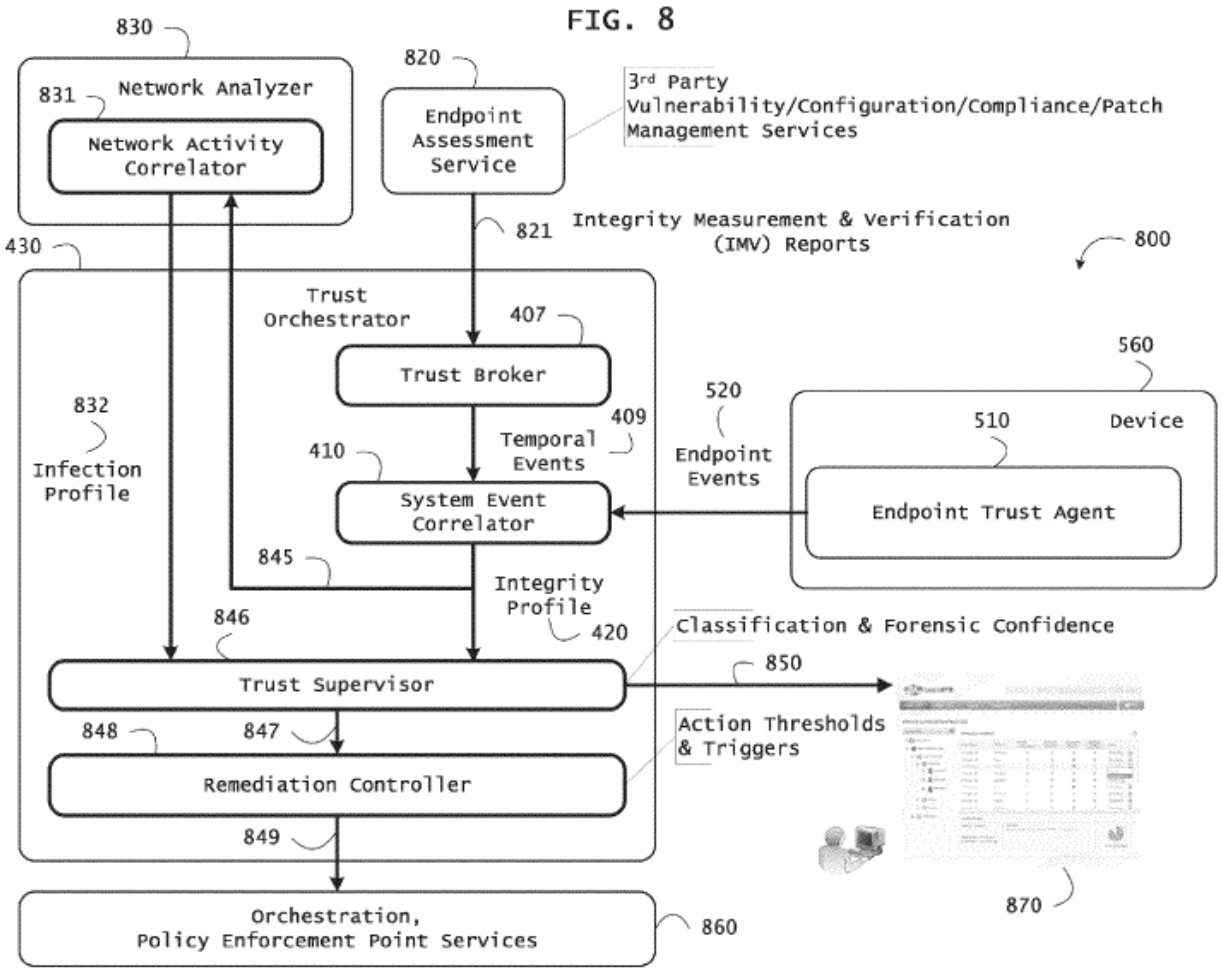


(*Id.* at Figure 6).

80. The computing environment of Figure 6 also includes a runtime monitor 620, an integrity processor 630, rulesets 631, an event correlation matrix 633, a system event correlator 410, and a risk correlation matrix 411. (*See id*. at 15:10-37). The "runtime monitor 620 [is] configured to register for and receive a plurality of raw events 611 from native machine instrumentation 511 on the device 560." (*Id*. at 15:12-19). An integrity processor 630, which resides on the host or computer being monitored, receives and processes information that may relate to the integrity of host computer, and generate warnings or alerts for the same. (*See id*. At 15:15-19, 17:17-27, 30:45-52, FIG. 6). The integrity processor 630 "may apply rulesets 631 to generate integrity alerts 632 and generate or populate an event correlation matrix 633". (*Id.* at

15:16-18). The event correlation matrix 633, which may be on the monitored device, "maps integrity warnings 634 to endpoint events 520". (*Id*. at 15:17-19). The system event correlator 410 correlates various events that may occur in the system to risks. (*See id*. at 14:56-59).

81. Another exemplary computing environment, shown in Figure 8, includes a network analyzer 830 and a trust supervisor 846. (*See id*. at 17:46-53). The network activity correlator 831, a component of the network analyzer 830, "monitors all network traffic and activities of device" and provides information "on forensic evidence of malicious network dialogs indicative of an infected victim or device D." (*Id*. at 20:40-44). "This evidence may be indicative of malicious activities." (*Id*. at 20:44-45). The trust supervisor 846 receives information for events relating to the monitored device and network, and processes the same to identify threats or vulnerability that compromise or affect the trust that may be assigned to the same. (*See id*. at 22:31-37, FIGS. 6, 8, 10). In some embodiments, the trust supervisor "correlate[s] events associated to a device 560 for the classification and identification of threats based on the forensic confidence of leading indicators." (*Id*. at 22:31-37).

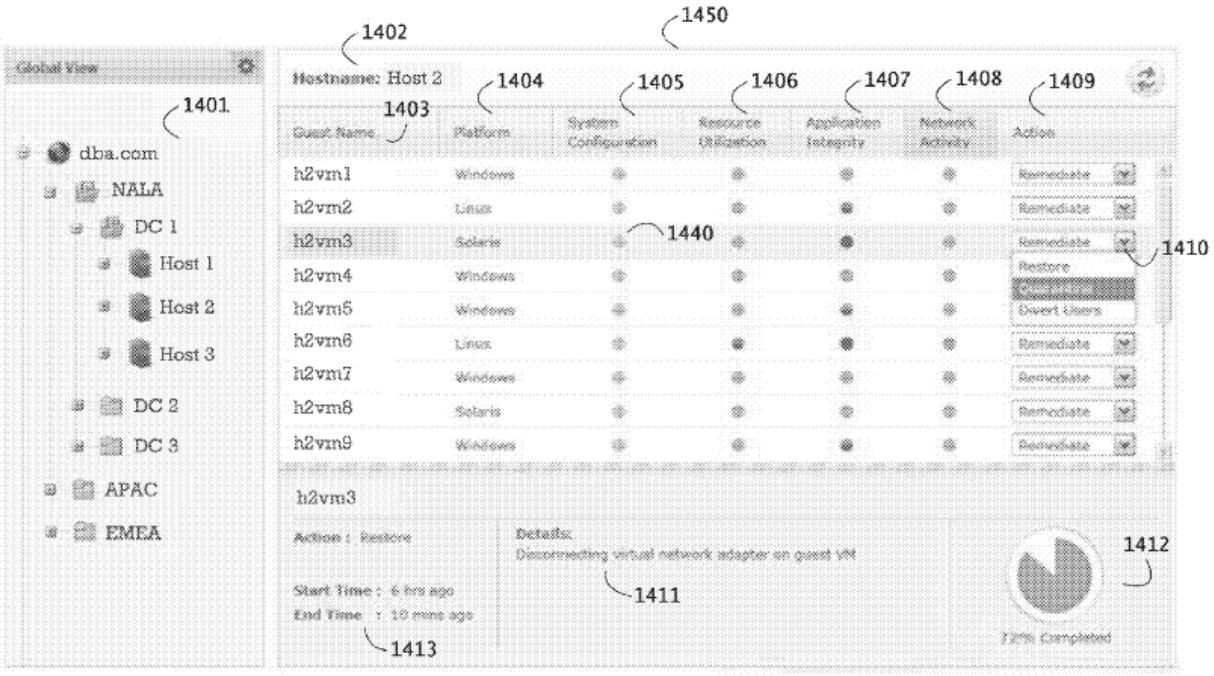


(*Id.* at Figure 8).

82. The status of the application's operational integrity, system configuration, resource, application integrity, and network activity is displayed in a graphical user interface (GUI) or dashboard for the administrator to view. (*See id.* at 7:8-12, FIGS 14-18).

1400

FIG. 14



(*Id.* at Figure 14).

B. PROSECUTION HISTORY

83. Prosecution began with a Restriction Office Action issues on April 11, 2014, to which Taasera responded by electing claims 5-9 for examination. (Ex. 1002 at pp. 142-148).

84. A non-final Office Action on July 17, 2014 rejected claims 5-9 under 35 U.S.C. §102(a)(1)(pre-AIA) and §103(a)(pre-AIA) as unpatentable as obvious in view of three patent references.[1] (*Id.* at p. 92-110).

85. After an Examiner interview, Taasera amended the claims into the form that issued. (*Id.* at pp. 69-74). Specifically, Taasera amended independent claim 5 as follows:

> 5. (Currently Amended) A method of providing real-time ~~runtime~~ operational integrity of an application on a native computing environment

[1] The Office Action cited U.S. Patent Publication Nos. 2008/0175266 to Alperovitch et al. (*see* Ex. 1010) and 2011/0145711 to Njemanze et al. (*see* Ex. 1011), and U.S. Patent No. 5,604,862 to Midgely et al. (*see* Ex. 1012). (Ex. 1002 at p. 108).

> ~~and a system using a computing platform comprising a network trust agent, an endpoint trust agent, an event and behavior correlation engine, and a trust orchestration server~~, the method comprising:
>
> monitoring, by a plurality of sensory inputs, one or more of network dialogs of the application, system operations initiated by the application, a runtime configuration of the application~~ activity, system configuration for a system the application is executing on~~, resource utilization by the application, and integrity of the application;
>
> generating real-time behavior based events for determining the real-time operational integrity of the application executing on the native computing environment which includes a network analyzer, an integrity processor an event correlation matrix, a risk correlation matrix, and a trust supervisor;
>
> correlating, by the event and risk correlation matrix, threat classifications ~~behavior correlation engine, risk~~ based on the temporal sequence of the generated real time behavior based events ~~monitoring~~; and
>
> displaying, in a plurality of runtime dashboards of an administrative console of the computing environment ~~platform~~, ~~real time~~ real-time status indications for operational integrity of the ~~system and~~ application.
>
> (*Id.* at pp. 71-72).

86. Taasera argued that the cited prior art failed to disclose the limitations of "generating real-time behavior based events for determining the real-time operational integrity of the application executing on the native computing environment which includes a network analyzer, an integrity processor, an event correlation matrix, a risk correlation matrix, and a trust supervisor." (*Id.* at pp. 77-78). Taasera also argued that the cited prior art failed to disclose "displaying, in a plurality of runtime dashboards of an administrative console of the computing environment, real-time status indications for operational integrity of the application." (*Id.* at p. 80).

87. Thereafter, a Notice of Allowance issued on December 24, 2014, which included a statement of allowable subject matter. Specifically, the Examiner stated that "none of the cited prior arts disclose the combination of limitations '**monitoring, by a plurality of sensory inputs, one or more of network dialogs of the application, system operations initiated by the**

application, a runtime configuration of the application resource utilization by the application, and integrity of the application; generating real-time behavior based events for determining the real-time operational integrity of the application executing on the native computing environment which includes a network analyzer, an integrity processor, an event correlation matrix, a risk correlation matrix, and a trust supervisor; correlating, by the event and risks correlation matrix, threat classifications based on the temporal sequence of the generated real-time behavior based events.'" (*Id.* at pp. 41-42, emphasis in original). The patent issued on March 24, 2015.

VII. LEVEL OF SKILL IN THE ART

88. Based on my knowledge and experience, I understand that a person of ordinary skill in the art ("POSITA") at the time of the earliest claimed priority date for the '948 patent would have had a bachelor's degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security. More education can supplement practical experience and vice versa.

89. I meet that definition of a person of ordinary skill in the art, and I did so on in and before May 2012. I also have worked with persons of ordinary skill in the art as part of my professional and academic experience, and I have an understanding of their skill level in and before May 2012.

VIII. OVERVIEW OF THE PRIOR ART

A. OVERVIEW OF "NETWORK INTRUSION DETECTION" BY MUKHERJEE ("MUKHERJEE")

90. "Network Intrusion Detection" is a research paper written by Biswanath Mukherjee, et al. ("Mukherjee"). The paper was published in 1994 and is directed towards the

state-of-the-art with respect to Intrusion Detection Systems. Mukherjee explains, "Intrusion detection systems (IDSs) are based on the beliefs that an intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable. Typically, IDSs employ statistical anomaly and rule based misuse models in order to detect intrusions." (Ex. 1005 at p. 1). In Mukherjee, "several host-based and network-based IDSs are surveyed, and the characteristics of the corresponding systems are identified." (*Id.*).

91. One of the IDS systems analyzed in Mukherjee is called Distributed Intrusion Detection System (DIDS).

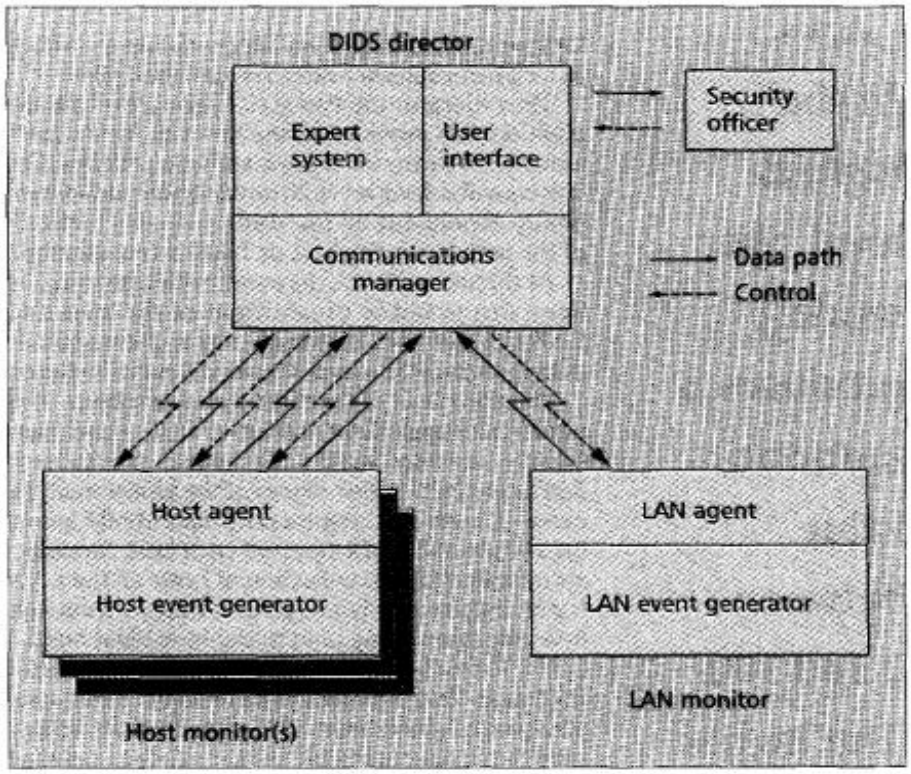
> The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDSs. The components of DIDS are the DIDS director, a single host monitor per host, and a single LAN monitor for each broadcast LAN segment in the monitored network. DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts. The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation.
> (Ex. 1005 at p. 11).

92. The DIDS director aggregates and evaluates this information received from host and LAN sensors to find patterns of suspicious behavior:

> The DIDS architecture performs information aggregation so that even if the activity of a network-wide user may not be suspicious on a single host, the aggregate behavior may be suspect or unpermitted. One of the strengths of this architecture is that it performs "accountability" by tying users with their actions.
> (*Id.*).

93. The host monitor collects information about an individual host, including events in audit records, such as "failed events, user authentications, changes to the security state of the system, and any network access such as rlogin and rsh." (*Id.* at p. 12).

94. The LAN monitor observes network traffic "in order to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns." (*Id.*). Figure 2 illustrates interactions between the DIDS director, host monitors, and LAN monitors:



■ **Figure 2.** *Communications architecture.*

(Ex. 1005 at Figure 2).

95. Finally, the DIDS director uses an expert system that is "responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole." (*Id.*).

96. The expert system taught by Mukherjee includes a six-level model of rules.

> The [Intrusion Detection Model] IDM describes the transformation from the distributed raw audit data to high-level hypotheses about intrusions and about the overall security of the monitored environment. This unified view of the distributed system simplifies the recognition of intrusive behavior that spans individual hosts. The model is the basis of the rule-base. The IDM consists of six layers, with each layer representing the result of a transformation performed on the data.
> (*Id.* at pp. 37-38).

97. The first layer represents raw audit records, represented in the format particular to their source (e.g., a LAN monitor, an operating system, a third-party auditing package). (*Id.* at p. 13). The second layer involves transforming those records to be "both syntactically and semantically independent of the source standard format for events." (*Id.*). The third layer introduces a subject (representing a user identification across potentially many hosts) and ceases to differentiate between individual hosts in a network. (*Id.*). The fourth layer introduces event context of two kinds: "temporal" (when events occur, especially when in proximity to other events) and "spatial" (where events occur on the network). (*Id.*). The fifth layer combines multiple events, together with their several associated contexts, in order to characterize threats, further "partitioned by the nature of the abuse and the nature of the target." (*Id.*). The sixth and final layer "produces a numeric value between one and 100 which represents the overall security state of the network. The higher the number, the less secure is the network. This value is a function of all the threats for all the subjects on the system." (*Id.*).

98. Mukherjee notes that apprising system administrators of relevant findings is an important part of intrusion detection systems: "Typically, an intrusion detection systems (IDS) alerts a system security officer (SSO) when it detects an attack." (*Id.* at p. 2). also discloses systems for displaying information to system administrators. Mukherjee discloses an example system for displaying information to system administrators, the "Information Security Officer's

Assistant (ISOA)," which provides display capabilities for the audit information and events collected, as well as analysis of that information. (*Id.* at pp. 30-31).

99. Mukherjee's teaches evaluating the integrity of a computer or a network based on traffic patterns and statistical anomalies. Although Mukherjee teaches this with reference to LAN systems, a POSITA would have understood that these techniques are also applicable to intrusion detection in a wide variety of networking environments, including networks in which computers have access to cloud computing resources. For example, Mukherjee's disclosures relating to host monitors apply just as well to computers connected by a LAN as to computers connected by a larger network (e.g., the Internet).

B. OVERVIEW OF US 7,451,488 TO COOPER ("COOPER")

100. U.S. Patent No. 7,451,488 to Geoffrey Cooper, et al. ("Cooper") is directed to "scanning for vulnerabilities on a continuous basis and interpreting the resulting traffic in the context of policy." (Ex. 1006 at 1:21-25). Cooper explains the basics of vulnerability scans:

> Vulnerability scanning technology is exemplified by commercial products such as the various scanners by Internet Security Systems, Inc. (ISS), the Cisco Scanner by Cisco Systems, Inc., and the Nessus Network Security Scanner. This technology examines the network configuration of hosts on the network by "probing" or "scanning" them using network traffic that is crafted to elicit a response. The response is interpreted and used to determine the configuration of the host that is scanned. Using various probing techniques, the technology is capable of:
>
> > scanning a range of Internet addresses to determine which addresses likely represent working hosts and which do not;
> >
> > scanning a range of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) port numbers within a given host to determine which applications are likely active on a given working host; and
> >
> > probing an application on a given host to determine whether it is susceptible to previously known attacks, known as "vulnerabilities."

(*Id.* at 1:55-64).

101. However, Cooper also explains that, at the time of Cooper's alleged invention, there were known weaknesses in this kind of scanning technology. One of these weaknesses is, "the sheer volume of information that it returns." (*Id.* at 2:10-17). This data also includes "many items that are already known to the network maintenance staff [for which] [t]he time spent removing [them] from a vulnerability scanner report significantly reduces its effectiveness." (*Id.*).

102. Cooper also identifies another problem with a vulnerability scanner is "its operation simulates that of an attacker." (*Id.* at 2:21-31). The reason this is a problem is because "… network monitoring technology, which is simultaneously monitoring the network is likely to detect a vulnerability scanner as an attacker and alert network maintenance staff that a strong and persistent attack is in progress." (*Id.*).

103. To solve these problems, Cooper discloses improvements to vulnerability scanning technology. For one such improvement, Cooper teaches that "[u]sing the network security policy's information, the output of the scanner is filtered, so that redundant information is removed about applications that are sanctioned by policy." (*Id.* at 2:56-63).

104. Furthermore, Cooper discloses that "the policy monitoring system is provided, through said network security policy, with a description of the intended action of the vulnerability scanner. This permits the monitoring system to report the scanner's operation as a normal usage pattern of the network, eliminating the confusion that can result when a vulnerability scan is monitored." (*Id.* at 2:64-3:2).

105. Cooper also discloses that the monitoring performed can be continuous.

> In one embodiment of the invention, vulnerability state information of a system can be derived from repeated events, referred to herein as continuous scanning. FIG. 3 is a schematic diagram showing a workflow of scanned vulnerability states, which are stored in a repository, and emitted as

> monitored events according to the invention. New vulnerability data is entered into the system and the output is a monitored event.
> (*Id.* at 9:56-64).

106. Another disclosure of Cooper is an "enterprise manager system user interface", which displays vulnerability events and network events in an interactive format.

> Vulnerability events are viewable through the enterprise manager system user interface. They are indistinguishable from other network events.
>
> In a live data page of the policy-based enterprise manager system, such as in Enterprise Manager UI's Live Data page, all vulnerability events generated during the specified query interval are collated under a specially named reporting element, referred to herein as the Vulnerability Scanners reporting element. In one embodiment of the invention, such Vulnerability Scanners reporting element is shown at the bottom of a set of top ten reporting elements as an 11th row. Thus, users can easily access such vulnerability events even if their volume is low.
>
> In some embodiments of the invention, the vulnerability event database may be accessed via an EM web interface 125 using database replication or a proxy request mechanism, e.g. HTTP proxy.
>
> Vulnerability alerts are viewable through the enterprise manager system user interface's alert page and they are handled equivalently to other network events. All alert management functions in the enterprise manager system user interface are applicable to vulnerability alerts.
> (Ex. 1006 at 15:21-43).

IX. GROUND 1: CLAIMS 1-5 ARE RENDERED OBVIOUS BY MUKHERJEE IN VIEW OF COOPER

107. Claims 1-5 of the '948 patent are rendered obvious by Mukherjee in view of Cooper, as explained in the sections below. I also explain that it would have been obvious for a POSITA to combine Mukherjee and Cooper.

A. Motivation to Combine Mukherjee and Cooper

108. A POSITA would have been motivated to combine Mukherjee and Cooper. Both systems are directed towards detecting and ameliorating vulnerabilities in network systems. Mukherjee discloses "Intrusion detection systems (IDSs) [that] are based on the beliefs that an

intruder's behavior will be noticeably different from that of a legitimate user and that many unauthorized actions are detectable." (Ex. 1005 at p. 1). Mukherjee explains that IDS's may incorporate "rules based on… known system vulnerabilities" for detecting intrusive behavior. (*Id.* at p. 5).

109. Cooper also analyzes vulnerabilities on a network system and specifically "scanning for vulnerabilities on a continuous basis and interpreting the resulting traffic in the context of policy." (Ex. 1006 at 1:21-25). Cooper is explicitly directed to working with "network monitoring technology." Cooper explains that when using traditional vulnerability scanners, "… network monitoring technology, which is simultaneously monitoring the network is likely to detect a vulnerability scanner as an attacker and alert network maintenance staff that a strong and persistent attack is in progress." (*Id.* at 2:23-26). Thus, recognizing that Cooper's active probing of a system for vulnerabilities can appear to be an attack, Cooper discloses a solution: "the policy monitoring system is provided, through said network security policy, with a description of the intended action of the vulnerability scanner. This permits the monitoring system to report the scanner's operation as a normal usage pattern of the network, eliminating the confusion that can result when a vulnerability scan is monitored." (Ex. 1006 at 2:64-3:2). A POSITA would have understood that IDS's that monitor networks are a "network monitoring technology."

110. A POSITA would have been motivated to combine Cooper's vulnerability scans, which are designed to operate effectively with network monitoring technology, into Mukherjee's network scanning IDS's. Mukherjee explains that:

> The DIDS architecture combines distributed monitoring and data reduction with centralized data analysis. This approach is unique among current IDSs. The components of DIDS are the DIDS director, a single host monitor per host, and a single LAN monitor for each broadcast LAN segment in the monitored network. ***DIDS can potentially handle hosts without monitors since the LAN monitor can report on the network activities of such hosts.***

> The host and LAN monitors are primarily responsible for the collection of evidence of unauthorized or suspicious activity, while the DIDS director is primarily responsible for its evaluation.
> (Ex. 1005 at p. 11, emphasis added).

111. Mukherjee also explains that IDSs, such as DIDS, are best used with other tools in combination.

> ***It is anticipated that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS.*** This will give the SSO the ability to actively respond to attacks against the system in real time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.
> (Ex. 1005 at p. 12, emphasis added).

112. A POSITA would have recognized the vulnerability scanner of Cooper as a "network-management tool" in the family of such tools anticipated by Mukherjee, particularly because Cooper is explicitly designed to work with a monitoring system (which as I explained, is similar to an IDS) and regards this integration as part of the advantages of its teachings (e.g., notifying the monitoring system of the intended scanning activity in order to report such activity as expected).

113. Thus, a POSITA would have recognized that the two disclosures of Mukherjee and Cooper are compatible by design. A POSITA would have been motivated by the disclosures above that indicate the systems perform best in combination with other systems. As cited previously in this section, Mukherjee recognizes that "known system vulnerabilities" (such as those identified by a vulnerability scanner) are already used in IDS systems. A POSITA would have been motivated to include vulnerability assessment data in Mukherjee's DIDS system.

114. Moreover, a POSITA would have recognized that vulnerability data is useful to Mukherjee's expert system for identifying intrusive behavior. For example, Mukherjee, in

defining a six-layer analysis system, explains that the fourth layer is for putting "the event in context." (Ex. 1005 at p. 13). Mukherjee describes that one kind of context is "relative importance of the source of events. That is, events related to a particular user, or events from a particular host, may be more likely to represent an intrusion…" (*Id.*). A POSITA would have known that vulnerability information provides this kind of context. That is, network traffic of a certain form to a vulnerable host is much more likely to be an intrusion than similar network traffic to a host that does not have that specific vulnerability. A POSITA would have been motivated to include the vulnerability information of Cooper to better identity context for events.

115. Similarly, Mukherjee also explains that the fifth layer of their intrusion model uses "[e]vents in context [combined] to create threats." (Ex. 1005 at p. 13). This layer includes "[s]uspicious acts [that] are events which, while not a violation of policy, are of interest to an IDS." (*Id.*). A POSITA would have known that network traffic to a vulnerable host is of more interest to the IDS than traffic to a non-vulnerable host, as it represents a greater likelihood for exploitation. A POSITA would have been motivated to include the vulnerability scans of Cooper to improve the identification of threats.

116. In addition to using vulnerability information to better contextualize and process events, Mukherjee would have been motivated to use Cooper's "vulnerability events" as events to the IDS. Mukherjee's first level of the six-level model receives "audit records provided by the host OS, by the LAN monitor, or ***by any third-party auditing package***." (Ex. 1005 at p. 13, emphasis added). Mukherjee specifically identifies system events (i.e., identified by records from the host OS) and network events (i.e., identified by the LAN monitor). Cooper discloses another kind of event: "[t]he scanner reports all application-level vulnerabilities as ***vulnerability events.***" (Ex. 1006 at 4:32-40, emphasis added).

117. A POSITA would have recognized that Cooper's "vulnerability events" could be included in Mukherjee's event processing. Cooper notes that "[i]n most respects, vulnerability events are handled in a similar fashion to network events." (Ex. 1006 at 7:64-8:6). Cooper explains in even more detail that a "vulnerability may also be ***mapped into a network event***…" (Ex. 1006 at 12:19-20, emphasis added). A POSITA would have understood this to mean that a vulnerability event can be translated into a type of network event. This is supported by Cooper's disclosures: "Vulnerability events are viewable through the enterprise manager system user interface. ***They are indistinguishable from other network events.***" (Ex. 1006 at 15:20-23, emphasis added).

118. As cited above, Mukherjee already discloses using data from "any third-party auditing package" as initial inputs to the first level of its IDM. (Ex. 1005 at p. 13). Thus, a POSITA would have known that the vulnerability events could be used either as a new kind of event, or as network events, and included in Mukherjee's first level of analysis. A POSITA would have been motivated to include this event information to more accurately identify intrusion by increasing the amount of available data and by reducing false positives, both of which are benefits to be found in a reference like Cooper. For example, Cooper explains that "the invention provides policy-based vulnerability scanning to ***minimize false positives***." (Ex. 1006 at 4:9-16, emphasis added).

119. Notably, Mukherjee's second level of analysis standardizes events such that a given event is "is both syntactically and semantically independent of the source standard format for events." (Ex. 1005 at p. 13). A POSITA would have understood that this standardization means that Mukherjee can insert events from alternative sources (such as Cooper) at level one without requiring significant changes to the processing in the other, higher layers.

120. Accordingly, based on the roadmaps in both Mukherjee and Cooper, a POSITA would have found it obvious to try Mukherjee's DIDS disclosures expanded by Cooper's vulnerability scanning disclosures. A POSITA would have recognized that these roadmaps reduce experimentation to a minimum and ensure a high likelihood of success.

121. A POSITA also would have been motivated to use Cooper specifically as opposed to any other general reference on vulnerability scanners. Cooper is designed to be unintrusive to other network monitoring systems. Cooper discloses that "the policy monitoring system is provided, through said network security policy, with a description of the intended action of the vulnerability scanner. This permits the monitoring system to report the scanner's operation as a normal usage pattern of the network, eliminating the confusion that can result when a vulnerability scan is monitored." (Ex. 1006 at 2:64-3:2).

122. Thus, I conclude that a POSITA would have been motivated to combine Mukherjee and Cooper.

B. Claim 1 is rendered obvious by Mukherjee in view of Cooper

123. Each of the limitations of Claim 1 of the '948 patent is rendered obvious by Mukherjee in view of Cooper as described below.

1. Limitation 1[a]

124. Limitation 1[a] of the '948 patent reads:

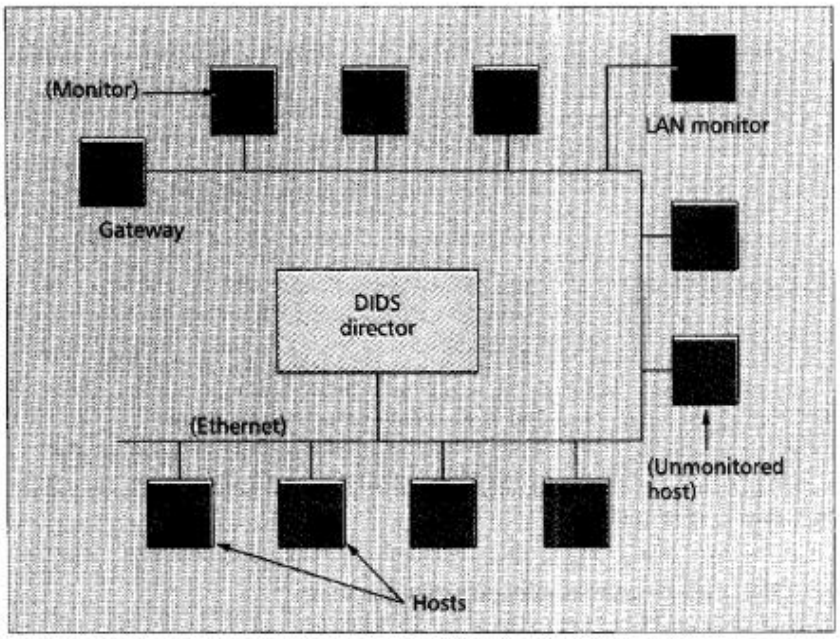
> *A method of providing real-time operational integrity of an application on a native computing environment, the method comprising*

125. I understand that limitation 1[a] is a "preamble" and that the preamble of claims is often not limiting. To the extent that limitation 1[a] is considered limiting, Mukherjee in view of Cooper discloses "a method of providing real-time operational integrity of an application on a native computing environment."

126. Mukherjee discloses real-time monitoring of hosts on a network. For example, Mukherjee explains that:

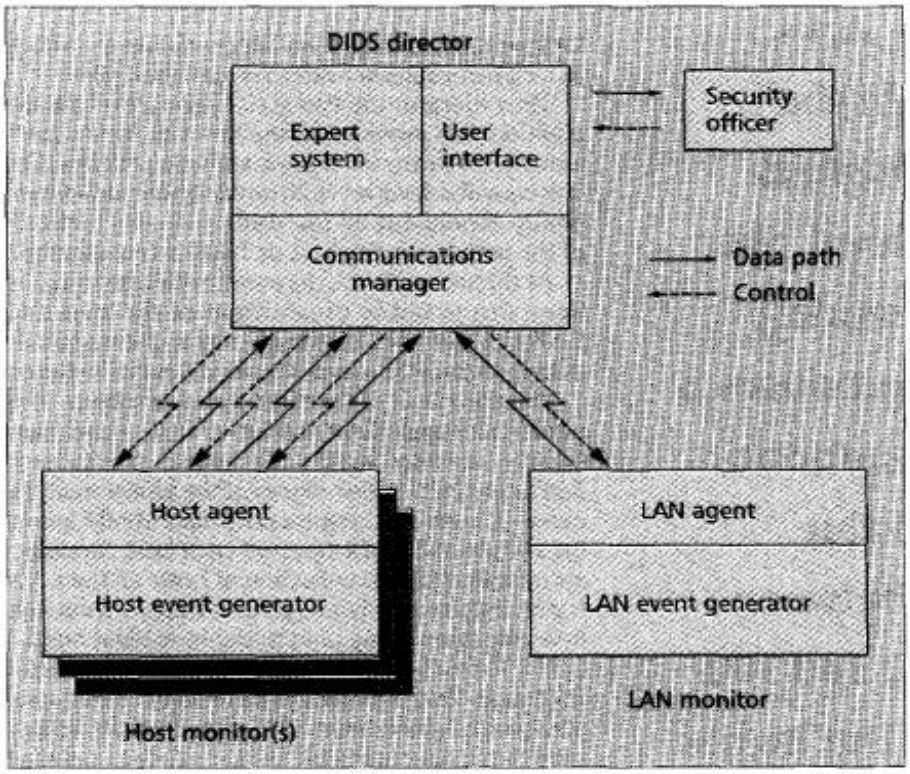
> The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole… The director's user interface allows the SSO interactive access to the entire system. ***The SSO can watch activities on each host, watch network traffic (by setting "wiretaps"), and request more specific types of information from the monitors***.
> (Ex. 1005 at p. 12, emphasis added).

127. A POSITA would have recognized that "watching activities on each host… [and] network traffic" refers to real-time monitoring, especially when using word such as "wiretaps". The real-time monitoring is also depicted in Figures 1 and 2, which are reproduced hereinbelow.



■ **Figure 1.** *DIDS target environment*

(Ex. 1005 at Figure 1).



■ **Figure 2.** *Communications architecture.*

(Ex. 1005 at Figure 2).

128. The '948 patent specification discusses "Evaluating Operational Integrity of Applications," by explaining that operational integrity monitoring for an application is based on "executable image profiles" that are then observed by a "process monitor 514 configured to observe local execution context of applications and services, … a socket monitor 513 configured to observe network activities of applications and services and a system monitor 512 configured to observe system and platform resources consumed by applications and services." (Ex. 1001 at 12:55-13:15). The data generated by these monitors is used by "an application integrity module 516 and a resource utilization module 515 configured to assess operational integrity based on rulesets 517." (*Id.*).

129. Based on these disclosures, a POSITA would have understood "operational integrity" to represent the ability of a system or application to operate reliably and securely in accordance with its intended purpose and rules.

130. Mukherjee in view of Cooper discloses a similar process. For example, Cooper discloses using vulnerability scans to get a baseline security profile of an application and then using runtime monitoring of the process, network, and system to determine if applications are compromised. For example, Cooper discloses monitoring applications through a vulnerability scanning process:

> Vulnerability scanning technology is exemplified by commercial products such as the various scanners by Internet Security Systems, Inc. (ISS), the Cisco Scanner by Cisco Systems, Inc., and the Nessus Network Security Scanner. This technology examines the network configuration of hosts on the network by "probing" or "scanning" them using network traffic that is crafted to elicit a response. The response is interpreted and used to determine the configuration of the host that is scanned. Using various probing techniques, the technology is capable of…
>
> > ***probing an application on a given host to determine whether it is susceptible to previously known attacks, known as "vulnerabilities."***
>
> (Ex. 1006 at 1:55-64, emphasis added).

131. Cooper explains further that "[o]ne embodiment of the invention adds vulnerability scanning capabilities to a policy-based monitor system and method to detect network services which contravene policy ***and to detect application-level vulnerabilities, e.g. a host running a vulnerable version of an open source http server, such as Apache.***" (Ex. 1006 at 4:10-16).

132. A POSITA would have known that public databases of known vulnerabilities exist and are used to identify vulnerable versions of operating systems and software applications. (*See* Section V.B.). Mukherjee explains that known vulnerabilities, such as those used by the vulnerability scanner disclosed by Cooper, are used in IDS systems to help identify intrusion.

> The Intrusion-Detection Expert System (IDES) developed at SRI International is a comprehensive system that uses complex statistical methods to detect atypical behavior, as well as an expert system that encodes known intrusion scenarios, ***known system vulnerabilities***, and the site-specific security policy [9-11].
> (Ex. 1005 at p. 5, emphasis added).

133. As I explain in detail below, Mukherjee specifically identifies monitoring applications, such as rsh, rlogin, and other "running processes". Accordingly, a POSITA would have understood that "known system vulnerability" includes application system vulnerabilities, such as those disclosed by Cooper as I discussed earlier in this limitation.

134. Mukherjee teaches "monitoring the real-time operational integrity of the application" by collecting events about applications and submitting them to a centralized component, the DIDS director, for analysis.

> The components of DIDS are the DIDS director, ***a single host monitor per host, and a single LAN monitor for each broadcast LAN segment in the monitored network***.
> (Ex. 1005 at p. 11, emphasis added).

> The host monitor consists of a host event generator (HEG) and a host agent. ***The HEG collects and analyzes audit records from the host's operating system***. The audit records are scanned for notable events.
> (*Id.*, emphasis added).

> Like the host monitor, the LAN monitor consists of a LAN event generator (LEG) and a LAN agent. The LEG is currently a subset of the ***NSM*** [17, 27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns.
> (*Id.* at p. 12, emphasis added).

135. Mukherjee explains that these records include application specific data. For example, Mukherjee discloses that the DIDS system incorporates HAYSTACK for the host scanning: "The HEG also tracks user sessions and report anomalous behavior aggregated over time

through user and group profiles and the integration of the HAYSTACK intrusion detection algorithm [5] into DIDS." (Ex. 1005 at p. 12).

136. Mukherjee further explains that HAYSTACK monitors applications for access from hackers. "The SSO may 'tag' particular security 'subjects' and 'objects' for special monitoring. This is analogous to setting an alarm to go off when a particular user-id is active, or when a particular file or program is accessed." (Ex. 1005 at p. 4).

137. Furthermore, Mukherjee discloses that these events are used in monitoring applications such as "rlogin" and "rsh".

> which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as ***rlogin and rsh***. These notable events are then sent to the director for further analysis.
> (Ex. 1005 at p. 12, emphasis added).

138. A POSITA would have understood that "rlogin" and "rsh" are applications that run in a "native computing environment." That is, "rlogin" and "rsh" are able to run natively on the system in question.

139. Moreover, Mukherjee discusses identifying security issues and attacks for "active objects" which include running processes:

> The targets of abuse are characterized as being either system objects or user objects and as being either passive or active. User objects are owned by nonprivileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, ***while active objects are essentially running processes***.
> (Ex. 1005 at p. 13).

140. A POSITA would have known that running processes are applications. Therefore, a POSITA would have understood Mukherjee's disclosures of collecting events related to

applications and analyzing them for security issues to teach the claimed "method of providing real-time operational integrity of an application on a native computing environment."

141. However, to the extent that Mukherjee does not explicitly teach "providing real-time operational integrity of an application on a native computing environment," it would have been disclosed by Mukherjee in view of Cooper. A POSITA would have been motivated to combine Cooper's analysis of application-specific vulnerabilities with Mukherjee's analysis of host and network events. As explained above, Mukherjee analyzes attacks on "running processes." It would have been obvious to use Cooper's per-application vulnerability information to more accurately identify these attacks and determine if the application is compromised.

142. Cooper discusses analyzing specific applications in detail. As discussed above, Cooper teaches analyzing vulnerabilities specific to an application, such as the web server "Apache." Cooper states: "[o]ne embodiment of the invention adds vulnerability scanning capabilities to a policy-based monitor system and method to detect network services which contravene policy ***and to detect application-level vulnerabilities, e.g. a host running a vulnerable version of an open source http server, such as Apache.***" (Ex. 1006 at 4:10-16).

143. Cooper identifies an exemplary vulnerability scanner called "Nessus." Using this scanner, or another, Cooper discusses analyzing application-level vulnerabilities.

> An exemplary scanner is the Nessus Security Scanner. However, it should be appreciated that other security scanners can be used. According to one embodiment of the invention, the scanner scans hosts as per a given policy. The scanner does not report on open ports. Instead, the policy-based monitor system performs such functionality, based on information in the same policy that controls the scanner. ***The scanner reports all application-level vulnerabilities as vulnerability events***. Scans may be scheduled by the user. (Ex. 1006 at 4:32-40, emphasis added).

144. According to Cooper, if vulnerabilities are detected, they are reported as "vulnerability events."

> Non-sanctioned applications, which are visible through open ports and services, are reported as monitored policy violations called "monitored events." ***Application-level vulnerabilities are reported by means of a special type of monitored event called a vulnerability event.*** The vulnerability event indicates the host where the vulnerability was detected, the protocol and service affected, as well as specific details of the vulnerability.
> (Ex. 1006 at 7:56-63, emphasis added).

145. Additionally, the scanning is on-going so that new vulnerabilities and new applications can be identified. "The vulnerability assessment system presents current vulnerability state of the infrastructure. When continuous scanning is used, the vulnerability assessment system provides changes to this state based on new vulnerabilities found or existing vulnerabilities removed. Throughout the process the policy adds context in terms specific to the customer applications allowing prioritized action to be taken to remediate problems." (Ex. 1006 at 3:6-15).

146. Moreover, Cooper discloses that the vulnerability events can behave like network events, similar to the ones disclosed in Mukherjee. In Cooper, the events are sent to a policy enforcement monitor.

> ***In most respects, vulnerability events are handled in a similar fashion to network events.*** They are summarized and rolled-up. Using a studio analyzer module, such as the SecurVantage™ Studio Analyzer, a vulnerability event can be drilled down from to get technical details of the actual vulnerability, similarly to protocol event details are obtained when drilling down through a network event. Vulnerability events with an assigned severity of CRITICAL generate alerts to all configured recipients of the policy-based monitor system alerts.
>
> ***When a vulnerability is discovered on a given host such vulnerability is reported as a vulnerability event and stored in the policy-based monitor system database 108, such as SecurVantage™ Monitor database, alongside the network event data.***
> (Ex. 1006 at 7:64-8:11, emphasis added).

147. A POSITA would have known that the network events reported to the policy monitor in Cooper are similar to the network events reported in Mukherjee to the DIDS IDS. To

the extent that Mukherjee's host and network data are not explicitly for application enforcement, it would have been obvious to a POSITA to incorporate Cooper's application-level vulnerability events into the Mukherjee disclosures.

148. Both Mukherjee and Cooper contemplate integration with references like each other. Mukherjee, for example, explains that IDS systems were known to take advantage of vulnerability information, as I already explained above. Mukherjee also discloses that the DIDS system was envisioned combined with other supportive systems.

> ***It is anticipated that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS.*** This will give the SSO the ability to ***actively respond to attacks against the system in real time***. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.
> (Ex. 1005 at p. 12, emphasis added).

149. Cooper is explicitly designed to work with a monitoring system (which as I explained, is similar to an IDS) and regards this integration as part of the advantages of its teachings. Cooper is designed to actively probe a system for vulnerabilities but recognizes that such probing can appear to be an attack. To solve this, Cooper discloses, "the policy monitoring system is provided, through said network security policy, with a description of the intended action of the vulnerability scanner. This permits the monitoring system to report the scanner's operation as a normal usage pattern of the network, eliminating the confusion that can result when a vulnerability scan is monitored." (Ex. 1006 at 2:64-3:2).

150. As explained in section IX.A., a POSITA would have been motivated to combine Mukherjee and Cooper. I incorporate this analysis into this section by reference.

151. Thus, Mukherjee in view of Cooper discloses "providing real-time operational integrity of an application on a native computing environment."

152. Mukherjee in view of Cooper discloses the remaining steps of the method as discussed in the paragraphs below.

153. Thus, I conclude that Mukherjee in view of Cooper discloses limitation 1[a].

2. Limitation 1[b]

154. Limitation 1[b] of the '948 patent reads:

> *monitoring, by a plurality of sensory inputs, one or more of network dialogs of the application, system operations initiated by the application, a runtime configuration of the application, resource utilization by the application, and integrity of the application;*

155. Mukherjee, or Mukherjee in view of Cooper, discloses limitation 1[b].

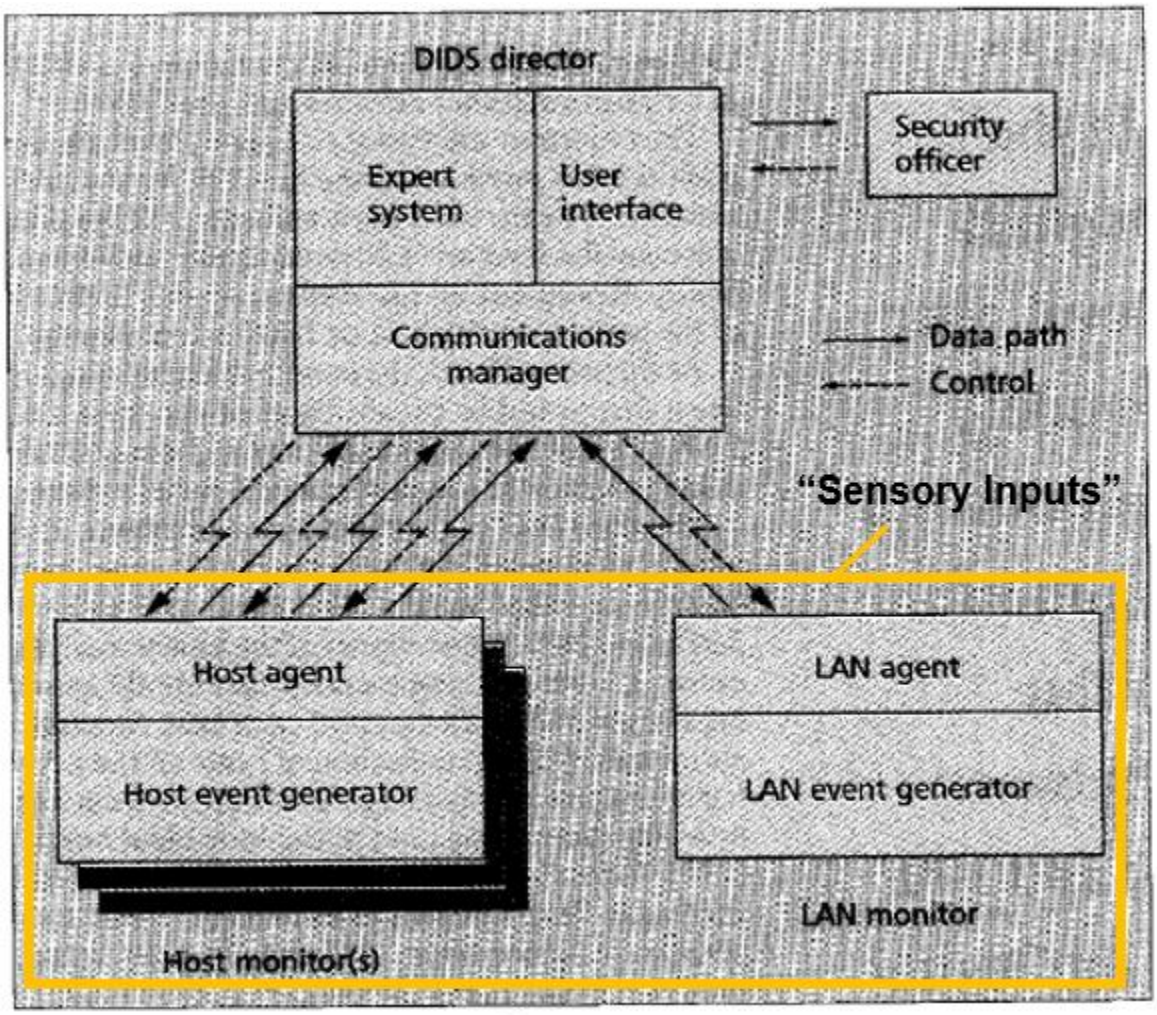
156. Mukherjee discloses "***monitoring by a plurality of sensory inputs***." For example, Mukherjee discloses both a "host monitor" and a "network monitor."

> The components of DIDS are the DIDS director, ***a single host monitor per host, and a single LAN monitor for each broadcast LAN segment in the monitored network***.
> (Ex. 1005 at p. 11, emphasis added).

> The host monitor consists of a host event generator (HEG) and a host agent. ***The HEG collects and analyzes audit records from the host's operating system***. The audit records are scanned for notable events.
> (Ex. 1005 at p. 11, emphasis added).

> Like the host monitor, the LAN monitor consists of a LAN event generator (LEG) and a LAN agent. The LEG is currently a subset of the ***NSM*** [17, 27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns.
> (Ex. 1005 at p. 12, emphasis added).

157. A POSITA would have understood the host event generator (HEG) and LAN monitor to teach the claimed "sensory inputs," because they monitor the data described in the remaining elements of this limitation, such as operating system audit records and network activities. These monitors are shown here annotated in Figure 2:



(Ex. 1005 at Figure 1, annotated).

158. A POSITA would have understood the claim language "one or more" to indicate that the system must monitor at least one of the claimed items. Mukherjee (and Mukherjee in view of Cooper) discloses at least two of the claimed items.

159. Mukherjee discloses "***monitoring… network dialogs of the application***." Mukherjee discloses monitoring network events, which a POSITA would have understood to be "network dialogs of the application." The '948 patent specification doesn't define "network dialogs", but a POSITA would have understood it to have a meaning similar to "network activity" or "network traffic", given this usage: "The ***network activity correlator*** 831 concurrently monitors all ***network traffic*** and activities of device D and dispatches infection profiles 832 based on forensic evidence of malicious ***network dialogs*** indicative of an infected victim (device D)." (Ex. 1001 at 20:40-44, emphasis added).

160. As I already explained above, Mukherjee discloses that DIDS includes both a network monitor and a host monitor but that ***both*** (not just the network monitor) track network events. For example, Mukherjee explains that the host monitor "… collects and analyzes audit records from the host's operating system. The audit records are scanned for notable events… [such as] any network access such as rlogin and rsh." (Ex. 1005 at pp. 11-12). As previously explained, rlogin and rsh are applications and a POSITA would have understood that Mukherjee discloses network dialogs of these applications.

161. Mukherjee also discloses that the network monitor tracks network events.

> Like the host monitor, the LAN monitor consists of a LAN event generator (LEG) and a LAN agent. The LEG is currently a subset of the NSM [17, 27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections, services used, and volume of traffic. ***The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns.*** (Ex. 1005 at p. 12, emphasis added).

162. Furthermore, Mukherjee in view of Cooper discloses "***monitoring… network dialogs of the application***." Cooper discloses that a vulnerability scanner connects to an application on a specific network port and determines from the network traffic whether the

application is vulnerable. "[F]or each vulnerability seen, the vulnerability may also be mapped into a network event as follows… The ***application protocol*** to which the vulnerability applies, e.g. http, is assigned to the service… The ***port where the vulnerability was found is assigned to the destination port.***" (Ex. 1006 at 12:12-13:26, emphasis added). As I explained in Section IX.A., a POSITA would have been motivated to combine Mukherjee and Cooper. Moreover, a POSITA would have recognized that Cooper's vulnerability scanning, which requires analyzing specific network ports, would be easily integrated into Mukherjee's network monitoring components such as the LAN monitor. A POSITA would have found Cooper's disclosures with respect to this limitation to be compatible with the techniques already disclosed by Mukherjee for "monitoring… network dialogs of the application." Because Cooper's scanning techniques can be implemented directly within components disclosed by Mukherjee, the POSITA would have a high expectation of success.

163. Mukherjee also discloses "***monitoring… system operations initiated by the application***." For example, Mukherjee discloses analyzing system operations such as notable events from the host's operating system.

> The host monitor consists of a host event generator (HEG) and a host agent. The HEG collects and analyzes ***audit records from the host's operating system***. The audit records are scanned for notable events. (Ex. 1005 at p. 11, emphasis added).

164. A POSITA would have known that audit records retained by the operating system include audit data not just from operating system actions but also from operations initiated by other applications (which, as I discussed in Section V.C., may include high-level user-facing software, services, or DLLs). Mukherjee explains that host audit records include application specific data. For example, Mukherjee discloses that the DIDS system incorporates HAYSTACK for the host scanning: "The HEG also tracks user sessions and report anomalous behavior aggregated over time

through user and group profiles and the integration of the HAYSTACK intrusion detection algorithm [5] into DIDS." (Ex. 1005 at p. 12).

165. Mukherjee further explains that HAYSTACK monitors applications for access from hackers. "The SSO may 'tag' particular security 'subjects' and 'objects' for special monitoring. This is analogous to setting an alarm to go off when a particular user-id is active, or when a particular file or program is accessed." (Ex. 1005 at p. 4).

166. Notably, a POSITA would have known that "system operations" refer to OS, or OS-like, operations. Thus, a POSITA would have recognized that these audit events, disclosed by Mukherjee, are system events that are triggered by applications. This is clear because Mukherjee discloses monitoring running processes:

> The targets of abuse are characterized as being either system objects or user objects and as being either passive or active. User objects are owned by nonprivileged users and/or reside within a non-privileged user's directory hierarchy. System objects are the complement of user objects. Passive objects are files, including executable binaries, while ***active objects are essentially running processes***.
> (Ex. 1005 at p. 13, emphasis added).

167. Thus, a POSITA would have recognized that Mukherjee's disclosures teach the claimed "monitoring… system operations initiated by the application".

168. Thus, I conclude that Mukherjee, or Mukherjee in view of Cooper, discloses limitation 1[b].

3. Limitation 1[c]

169. Limitation 1[c] of the '948 patent reads:

> *generating real-time behavior based events for determining the real-time operational integrity of the application executing on the native computing*

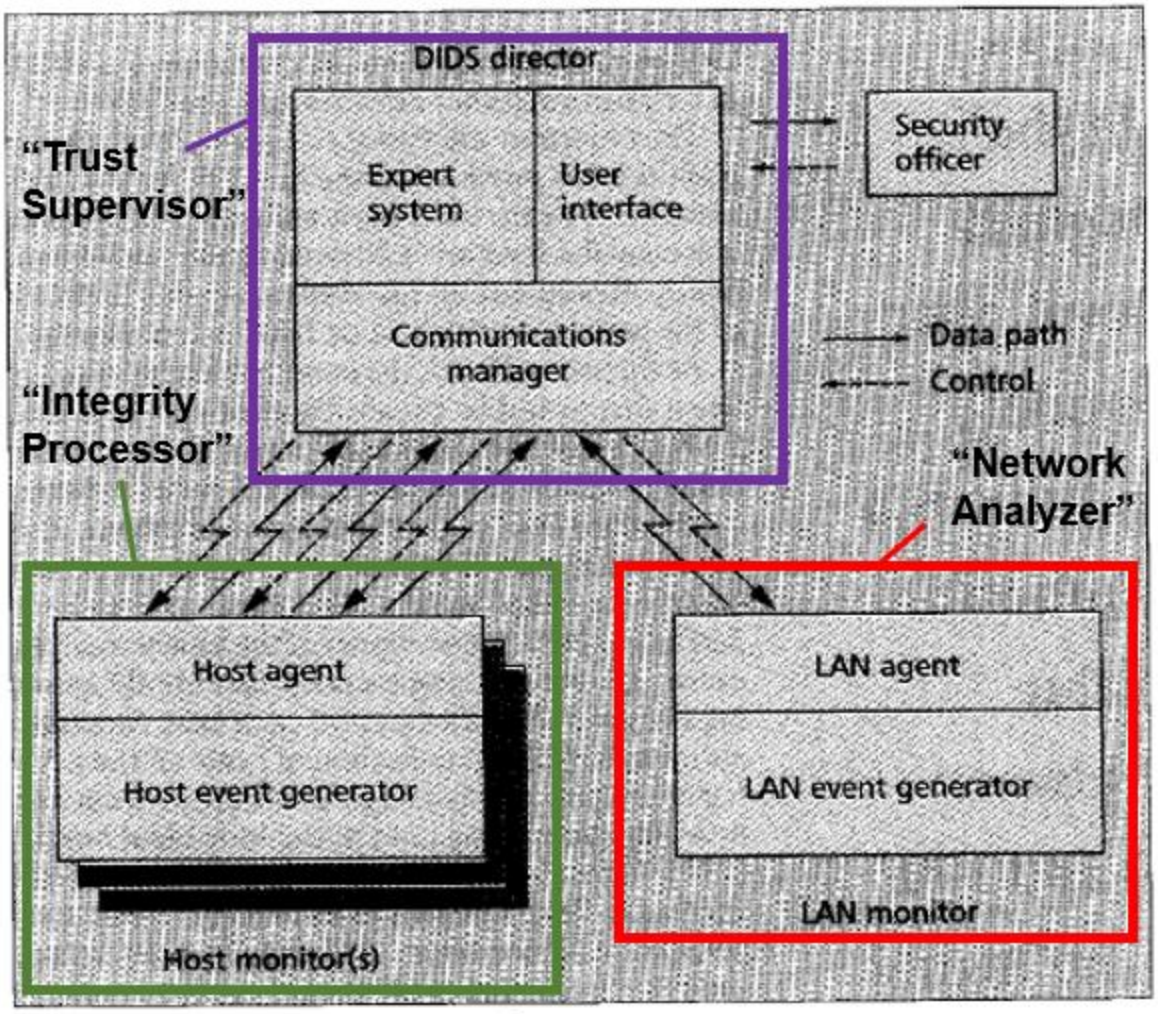
environment which includes a network analyzer, an integrity processor, an event correlation matrix, a risk correlation matrix, and a trust supervisor;

170. Mukherjee in view of Cooper discloses limitation 1[c].

171. Mukherjee discloses "generating real-time behavior based events…" For example, Mukherjee explains that DIDS operates in real-time.

> The expert system is responsible for evaluating and reporting on the security state of the monitored system. It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole… The director's user interface allows the SSO interactive access to the entire system. ***The SSO can watch activities on each host, watch network traffic (by setting "wiretaps"), and request more specific types of information from the monitors***.
> (Ex. 1005 at p. 12, emphasis added).

172. A POSITA would have recognized that "watch[ing] activities on each host, [and] watch[ing] network traffic" refers to real-time monitoring, especially when using words such as "wiretaps". The real-time monitoring is also depicted in Figure 2, which is reproduced and annotated hereinbelow:



(Ex. 1005 at Figure 2, annotated).

173. Additionally, Mukherjee explains that behavior-based events are produced in real-time. As Mukherjee explains, "In DIDS, the host and LAN monitors report any 'interesting' events, which may possibly lead to intrusive activity, to a centrally-located DIDS director." (Ex. 1005 at p. 11). These events are detected on hosts wherein the events "include, among others, failed events, user authentications, changes to the security state of the system, and any network access such as rlogin and rsh." (*Id.* at p. 12). Events also include "activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns."

(*Id.*). A POSITA would have recognized all of these as behavior-based events because each of them is indicative of some kind of user behavior.

174. Mukherjee also discloses "generating real-time behavior based ***events for determining the real-time operational integrity of the application executing on the native computing environment***…" As discussed with reference to limitations 1[a] and 1[b], Mukherjee discloses monitoring applications. To the extent that monitoring applications is not explicit, Mukherjee in view of Cooper does explicitly monitor applications. I incorporate my analysis of these two limitations into this section by reference.

175. Mukherjee also explains that the event monitoring is used to monitor any kind of intrusive behavior, including intrusions that alter application state. According to Mukherjee's teachings, the DIDS system:

> … considers the threats to the network and the hosts connected to it. Events in context are combined to create threats. The threats are partitioned by the nature of the abuse and the nature of the target. ***In other words, what is the intruder doing, and what is he doing it to?*** Abuses are divided into attacks, misuses, and suspicious acts. Attacks represent abuses in which the state of the machine is changed. That is, ***the file system*** or ***process state is different after the attack than it was prior to the attack.***
> (Ex. 1005 at p. 13).

176. A POSITA would have known that a different "process state… after the attack" indicates that the operational integrity of one or more user applications (such as rlogin and rsh) may have been impaired because an attack has changed them. That is, the application may no longer be able to operate reliably and securely in accordance with its intended purpose and rules. Moreover, changes to the file system because of an attack could indicate that the operating system software application has lost operational integrity (e.g., through injection of malicious code, modification of operating parameters, or other compromise of the application's intended state).

177. In addition, Mukherjee's disclosures of monitoring the state of "the file system or process state" indicate that the monitoring is of applications executing on the machine, that is, in a "native computing environment," because, as a POSITA would have understood, the file system and the state of a process are concepts relevant to a "native computing environment." (*See* my analysis of limitation 1[a] with reference to the term "native computing environment.").

178. The language of this claim is not clear about the antecedent of the phrase "which includes…". In my analysis below, I have interpreted "which includes" to refer to "the native computing environment."

179. Mukherjee further discloses "… the native computing environment which includes a network analyzer." Mukherjee discloses a LAN monitor that processes network-based events.

> In DIDS, the host and ***LAN monitors report any "interesting" events,*** which may possibly lead to intrusive activity, to a centrally-located DIDS director.
> (Ex. 1005 at p. 11, emphasis added).
>
> The components of DIDS are the DIDS director, a single host monitor per host, and a ***single LAN monitor for each broadcast LAN segment*** in the monitored network.
> (Ex. 1005 at p. 11).
>
> Like the host monitor, the ***LAN monitor consists of a LAN event generator (LEG) and a LAN agent***. The LEG is currently a subset of the NSM [17,27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections services used, and volume of traffic. ***The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns***.
> (Ex. 1005 at p. 12).
>
> …each layer representing the result of a transformation performed on the data [e.g., from a lower level]. ***The objects at the first (i.e., lowest) level are the audit records provided by the host OS, by the LAN monitor, or by any third-party auditing package***. The objects at this level are both syntactically and semantically dependent on the source. At this level, all of the activity on the host or LAN is represented. At the second level, the event (which has already been discussed in the context of the host and LAN monitor) is both

> syntactically and semantically independent of the source standard format for events.
> (Ex. 1005 at p. 13).

180. A POSITA would have recognized Mukherjee's "LAN monitor" to teach the same or substantially similar concept as the claimed "network analyzer" as disclosed by the '948 patent, because such a monitor "observe[s]," "monitor[s]," and "reports on" network traffic. (Ex. 1005 at p. 12).

181. Cooper also discloses a network analyzer. Cooper explains that vulnerability scans use network scanning to create inventories of devices that can be attacked, determine which network ports (e.g., TCP and UDP ports) are in use, and "probe" them over the network to determine application-level vulnerabilities:

> Vulnerability scanning technology is exemplified by commercial products such as the various scanners by Internet Security Systems, Inc. (ISS), the Cisco Scanner by Cisco Systems, Inc., and the Nessus Network Security Scanner. ***This technology examines the network configuration of hosts on the network by "probing" or "scanning" them using network traffic that is crafted to elicit a response. The response is interpreted and used to determine the configuration of the host that is scanned***. Using various probing techniques, the technology is capable of:
>
> > ***scanning a range of Internet addresses*** to determine which addresses likely represent working hosts and which do not;
> >
> > ***scanning a range of Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) port numbers*** within a given host to determine which applications are likely active on a given working host; and
> >
> > ***probing an application on a given host*** to determine whether it is susceptible to previously known attacks, known as "vulnerabilities."
>
> (Ex. 1006 at 1:55-64).

182. As Cooper explains in this citation, the scanning of the vulnerability scanner uses network traffic to elicit a response. Vulnerabilities are inferred from the response. Cooper further

explains that "[t]he scanner reports all application-level vulnerabilities as ***vulnerability events.***" (Ex. 1006 at 4:32-40). A POSITA would have understood these vulnerability events to be "*generat[ed] real-time behavior based events for determining the real-time operational integrity of the application*" because the events are generated by the vulnerability scanner in real-time based on the responses, which are examples of network behavior, and indicate risks to the application that could be exploited.

183. I have already explained in Section IX.A. that a POSITA would have been motivated to combine Mukherjee and Cooper and I incorporate this analysis into this section by reference. As I explained in that section, a POSITA would have been motivated to include Cooper's vulnerability events into Mukherjee's DIDS system.

184. Mukherjee also discloses "… the native computing environment which includes… an integrity processor…" Mukherjee discloses a host monitor that tracks system events.

> In DIDS, the ***host*** and LAN ***monitors report any "interesting" events***, which may possibly lead to intrusive activity, to a centrally-located DIDS director.
> (Ex. 1005 at p. 11).
>
> The components of DIDS are the DIDS director, ***a single host monitor per host,*** and a single LAN monitor for each broadcast LAN segment in the monitored network.
> (*Id.*).
>
> The host monitor consists of a host event generator (HEG) and a host agent. ***The HEG collects and analyzes audit records from the host's operating system. The audit records are scanned for notable events.***
> (*Id.*).

185. As I explained earlier in this section, a POSITA would have understood that Mukherjee's disclosure of the HEG monitoring host events (such as file system events and audit records) teaches a concept similar to "an integrity processor," because these events indicate when changes occur that could compromise the integrity of the system or the applications running on

the system. For example, changes to the file system could indicate that the operating system software application has lost operational integrity (e.g., through injection of malicious code, modification of operating parameters, or other compromise of the application's intended state).

186. Mukherjee discloses "… the native computing environment which includes… an event correlation matrix…" Mukherjee discloses that the monitoring is in a "native computing environment." (*See* my analysis in limitation 1[a] with respect to a POSITA's understanding of the term "native computing environment"). For example, Mukherjee explains that "The [Host Event Generator component of the host monitor] collects and analyzes audit records from the host's operating system." (Ex. 1005 at p. 11). A POSITA would have understood events from the host operating system to be "native" rather than non-native events. This is consistent with the specification of the '948 patent which explains that "In embodiments, the native machine instrumentation 511 ***can represent any event subscription, callback, notification mechanism provided by the supported operating system*** (OS) on the device 560." (Ex. 1001 at 13:43-56, emphasis added).

187. Mukherjee also discloses an event correlation matrix within this environment. The specification of the '948 patent describes the event correlation matrix as "…map[ing] integrity warnings 634 to endpoint events 520…" (Ex. 1001 at 15:10-19). Likewise, Mukherjee discloses mapping events to warnings and other messages related to system integrity in the "fourth layer" of the Intrusion Detection Model.

> The fourth layer of the model introduces the event in context. There are two kinds of context: temporal and spatial. As an example of temporal context, behavior which is unremarkable during standard working hours may be suspicious during off-hours [35]. The IDM, therefore, allows for the application of information about wall-clock time to the events it is considering. Wall-clock time refers to information about the time of day, weekdays versus weekends and holidays, as well as periods when an increase in activity is expected. In addition to the consideration of external

> temporal context, the expert system uses time windows to correlate events occurring in temporal proximity. This notion of temporal proximity implements the heuristic that a call to the UNIX who command followed closely by a login or logout is more likely to be related to an intrusion than either of those events occurring alone. ***Such related patterns of behavior are also referred to as attack signatures*** [23,24]."
> (Ex. 1005 at p. 13).

188. A POSITA would have known that "attack signatures," in the context of IDS, refers to patterns of behavior known to be related to attacks. A POSITA would have understood the fourth layer of the IDM to be an "event correlation matrix." Thus, Mukherjee discloses "mapping integrity warnings to endpoint events."

189. Moreover, Mukherjee discloses "… the native computing environment which includes… a risk correlation matrix…" The specification of the '948 patent explains that a "risk correlation matrix" comprises "grids that represent an exemplary dynamic model of measurement and identification based on clustering and classification of independent endpoint events (alerts) to generate system warnings that may be mapped ***to warning categories or classes***." (Ex. 1001 at 12:46-54, emphasis added). The '948 patent also discloses that "system warnings and threat classification [are] produced by the risk correlation matrix 411." (*Id.* at 14:51-15:2).

190. Mukherjee explains that, in the DIDS system, the fifth layer of the Intrusion Detection Model maps events (within their context produced in level four) to threats.

> The fifth layer of the model considers the threats to the network and the hosts connected to it. ***Events in context are combined to create threats.*** The threats are partitioned by the nature of the abuse and the nature of the target. ***In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into attacks, misuses, and suspicious acts***.
> (Ex. 1005 at p. 13, emphasis added).

191. A POSITA would have known that these attack categories represent different types of threats and that the fifth layer of the DIDS IDM teaches a "risk correlation matrix." This is consistent with the '948 patent's use of "threat categories or classes" as cited above.

192. Finally, Mukherjee discloses "… the native computing environment which includes… a trust supervisor." The '948 patent explains that "[t]he trust supervisor 846 uses an event processor 1006, an event correlator 1007, a threat classifier 1008, an event correlation matrix 1013 and a rules repository 1012 to correlate events associated to a device 560 for the classification and identification of threats based on the forensic confidence of leading indicators." (Ex. 1001 at 22:31-36).

193. Mukherjee explains that DIDS has a "director" that receives reports from the monitors and, from them, makes the inferences about system security.

> The DIDS director consists of three major components… a communications manager, an expert system, and a user interface… The expert system is responsible for evaluating and reporting on the security state of the monitored system. ***It receives the reports from the host and the LAN monitors, and, based on these reports, it makes inferences about the security of each individual host, as well as the system as a whole***… The director's user interface allows the SSO interactive access to the entire system. The SSO can watch activities on each host, watch network traffic (by setting "wiretaps"), and request more specific types of information from the monitors.
> (Ex. 1005 at p. 12, emphasis added).

194. Moreover, the DIDS director comprises the expert system. The different levels of the IDM discussed above are part of the expert system. Mukherjee explains: "DIDS utilizes a rule-based (or production) expert system written in CLIPS. The expert system uses rules derived from a hierarchical Intrusion Detection Model (IDM)." (Ex. 1005 at p. 12). This is consistent with the descriptions of the '948 patent that "[t]he trust supervisor 846 uses an event processor 1006, an event correlator 1007, a threat classifier 1008, an event correlation matrix 1013 and a rules repository 1012 to correlate events associated to a device 560 for the classification and identification of threats based on the forensic confidence of leading indicators." (Ex. 1001 at

22:31-36). A POSITA would have understood Mukherjee's DIDS director to teach the claimed "trust supervisor."

195. Thus, I conclude that Mukherjee in view of Cooper discloses limitation 1[c].

4. Limitation 1[d]

196. Limitation 1[d] of the '948 patent reads:

> *correlating, by the event and risk correlation matrix, threat classifications based on the temporal sequence of the generated real-time behavior based events; and*

197. Mukherjee and Cooper each disclose limitation 1[d].

198. I have already explained with respect to limitation 1[c] that Mukherjee discloses event and risk correlation matrices. I also explained that event correlation matrices map events to warnings, and risk correlation matrices map events and warnings to threat categories and classes. I incorporate this analysis from 1[c] into this limitation by reference.

199. Mukherjee also discloses performing the threat classification "based on the temporal sequence of the generated real-time behavior based events." Mukherjee explains that the event processing of the fourth layer of the IDM, which as I already explained, a POSITA would have recognized as an event correlation matrix, uses temporal proximity to correlate events.

> In addition to the consideration of external temporal context, the expert system ***uses time windows to correlate events occurring in temporal proximity*.** This notion of temporal proximity implements the heuristic that a call to the UNIX who command followed closely by a login or logout is more likely to be related to an intrusion than either of those events occurring alone. Such related patterns of behavior are also referred to as attack signatures [23,24].
> (Ex. 1005 at p. 13, emphasis added).

200. Mukherjee also teaches that level five, which I have already explained would have been understood by a POSITA as a risk correlation matrix that produces threat classifications, relies on the output of level four; namely the events "in context."

> The fifth layer of the model considers the threats to the network and the hosts connected to it. ***Events in context are combined to create threats.*** The threats are partitioned by the nature of the abuse and the nature of the target. ***In other words, what is the intruder doing, and what is he doing it to? Abuses are divided into attacks, misuses, and suspicious acts***. (Ex. 1005 at p. 13, emphasis added).

201. Because the fifth layer is dependent on the fourth layer, the results of the fifth layer are also based on the temporal proximity discussed above.

202. Mukherjee is interested in correlating events based on temporal sequence. For example, Mukherjee explains that "behavior which is unremarkable during standard working hours may be suspicious during off-hours." (Ex. 1005 at p. 13). Thus, a POSITA would have understood these disclosures of Mukherjee to teach "correlating, by the event and risk correlation matrix, threat classifications based on the temporal sequence of the generated real-time behavior events."

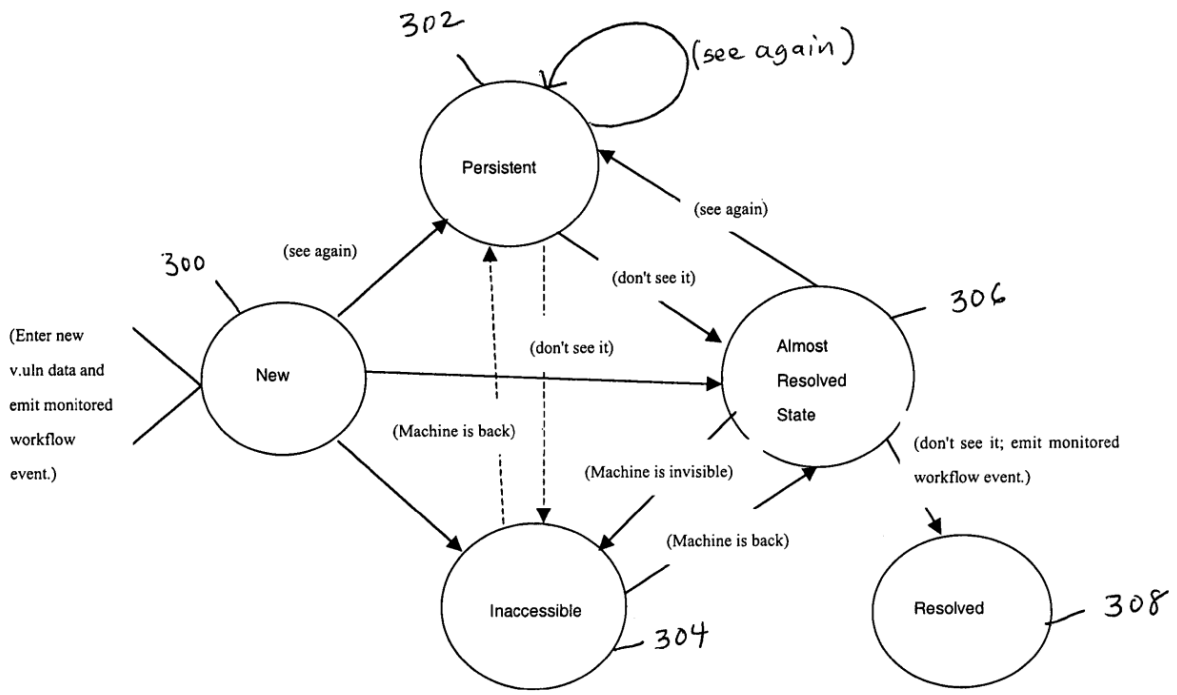
203. Cooper also discloses "correlating, by the event and risk correlation matrix, threat classifications based on the temporal sequence of the generated real-time behavior based events." Specifically, Cooper discloses "continuous scanning" to produce states that vary based on when and how often an event is detected: "When continuous scanning is used, the vulnerability assessment system provides changes to this state based on new vulnerabilities found or existing vulnerabilities removed." (Ex. 1006 at 3:7-10). Example states are "new" (when seen for the first time) and "persistent" (when seen multiple times). Cooper explains:

> Referring to FIG. 3, new vulnerability data enters the system at state new 300 and a monitored event is emitted. ***If such vulnerability data is seen again, the state becomes persistent 302. If it is not seen again, it becomes almost resolved 306***. If the machine on which the vulnerability is detected is no longer visible on the network, the state becomes inaccessible 304 and

a monitored event is emitted. ***From persistent 302, if the vulnerability is detected again, the state remains persistent 302. From almost resolved 306, if a vulnerability is seen again, the stated changes to persistent 302. If the vulnerability is not seen after an amount of time, the state becomes resolved 308 and a monitored event is emitted.*** If the vulnerability is from a machine which ceases to be visible on the network, the state becomes inaccessible 304. From the inaccessible state, if a machine returns from being invisible from the almost resolved state 306, the state returns to the almost resolved state 306. If a machine returns from being invisible from a persistent state 302, the state returns to the persistent state 302.
(Ex. 1006 at 10:21-39, emphasis added).

204. See also Cooper's Figure 3, which I have reproduced hereinbelow:

Figure 3



(Ex. 1006 at Figure 3).

205. A POSITA would have recognized that Cooper's disclosures are compatible with those of Mukherjee and are in the same field of endeavor, including with respect to limitation 1[d].

In particular, Cooper's "continuous scanning" has a similar goal to that of Mukherjee: performing event correlations in which the temporal context is relevant. For example, a POSITA would have recognized that network traffic directed to a host with a one-time vulnerability may be unimportant. But exploitative behavior directed to a long-standing vulnerability may indicate that an attacker has learned, and is using, the vulnerability to exploit the system. (*See*, e.g., Section V.B. with respect to known vulnerabilities).

206. A POSITA would have understood that repeated exploitation of long-standing vulnerabilities, as discussed above, incorporate a "temporal sequence of monitored network events." Thus, a POSITA would have found it obvious to combine Cooper's teachings incorporating temporal analyses into the event correlation matrix and the risk correlation matrix threat classifications of Mukherjee.

207. Thus, I conclude that Mukherjee and Cooper both disclose limitation 1[d].

5. Limitation 1[e]

208. Limitation 1[e] of the '948 patent reads:

> *displaying, in a plurality of runtime dashboards of an administrative console of the computing environment, real-time status indications for operational integrity of the application.*

209. Mukherjee and Cooper each disclose limitation 1[e].

210. Mukherjee discloses "displaying, in… an administrative console of the computing environment, real-time status indications for operational integrity of the application." Mukherjee explains that the DIDS director includes a user-interface for permitting the System Security Officer (SSO) views into and control over the IDS.

> The director's user interface allows the SSO interactive access to the entire system. The SSO can watch activities on each host, watch network traffic

> (by setting "wiretaps"), and request more specific types of information from the monitors.
> (Ex. 1005 at p. 12).

211. Mukherjee also discloses "a plurality of runtime dashboards." The '948 patent specification explains that dashboards are used for purposes such as "displaying data related to operational integrity, resource utilization, application integrity, and network activity." (Ex. 1001 at 26:37-46). Mukherjee discloses displaying data related to network traffic, which a POSITA would have understood to be "network activity." Mukherjee also discloses displaying data related to activities on each host, which a POSITA would have understood to be related to "application integrity." Thus, Mukherjee's disclosures of such data display concord with the data types and purposes for the dashboards taught in the '948 patent, and they also disclose "a plurality of runtime dashboards."

212. Cooper also discloses "displaying, in a plurality of runtime dashboards of an administrative console of the computing environment, real-time status indications for operational integrity of the application." Cooper discloses displaying real-time vulnerability information for individual systems along with the risks associated with them, which a POSITA would have understood to represent "operational integrity" of those systems.

> The vulnerability event data can also be integrated into the PDD 114 accessible through the policy-based enterprise manager system user interface, such as Enterprise Manager UI. Each network object page includes a link to vulnerability information about that network object. Unlike the remainder of the PDD 114, such vulnerability information is generated dynamically by querying the vulnerability event database. ***The vulnerability information is rolled up by host IP address and vulnerability severity, as identified by the outcome component assigned to the events***.
> (Ex. 1006 at 8:45-54, emphasis added).
>
> In the PDD 114, each network object page includes a link to vulnerability information pertaining to the network object. This link invokes a JSP with the appropriate parameters so as to execute a query against the vulnerability events in the database. ***The JSP constructs an HTML page(s) with***

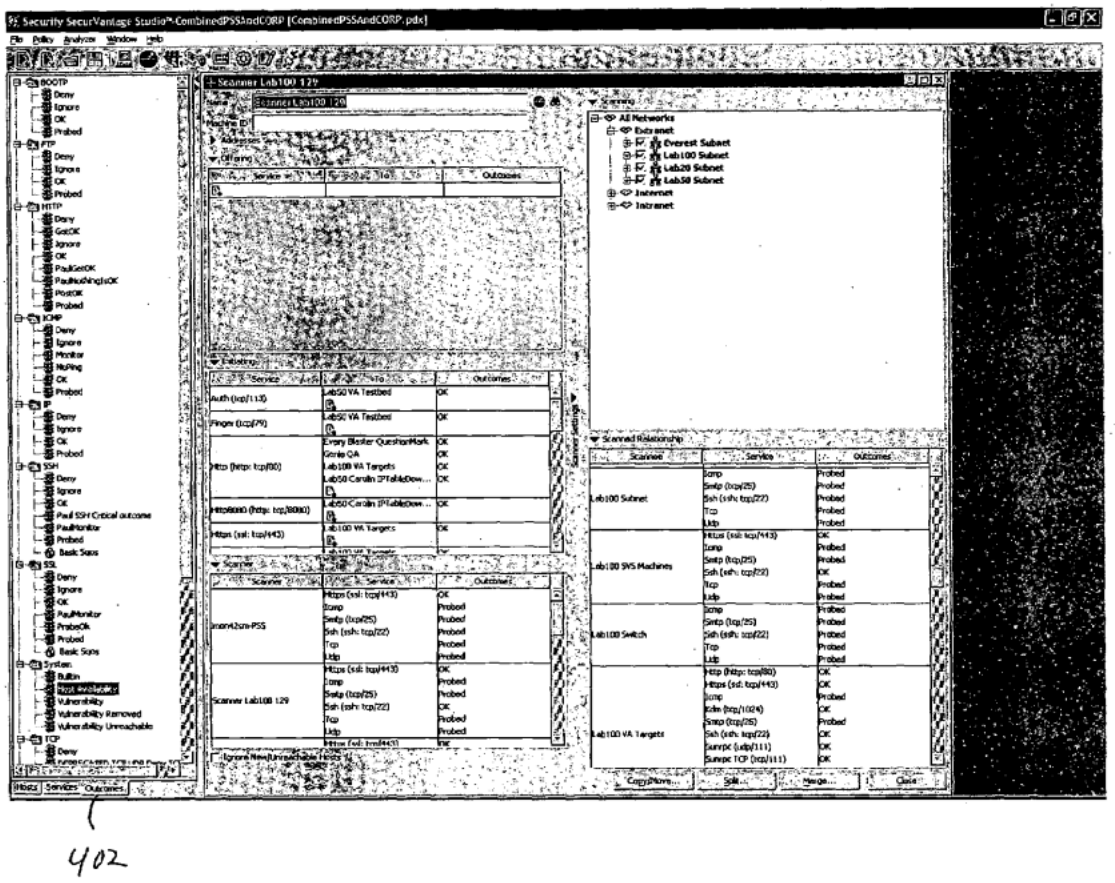
> ***vulnerability information sorted by the IP address of the target host. For each target host, a list of vulnerabilities that affect it is displayed***. Clicking on each individual vulnerability event provides the vulnerability details information. A PDD 114 accessed through an enterprise manager system provides a view of vulnerability information for an entire policy domain. A PDD 114 accessed through the policy-based monitor system 100 provides visibility only to the hosts scanned by the VA Server on that monitor.
> (Ex. 1006 at 15:47-61, emphasis added).

213. Cooper provides an example of monitoring the event data from a "studio module". In this example, the system displays outcomes, policies, and other relationships.

> FIG. 4 is an example view from the studio module according to the invention. The outcomes tab 402 is selected in the panel window on the left, showing possible outcomes of events. From the interactive screen, policy for the scanner and scanning relationships are further defined.
> (Ex. 1006 at 16:26-30).

214. See also, Cooper Figure 4, which I have reproduced hereinbelow:

Figure 4



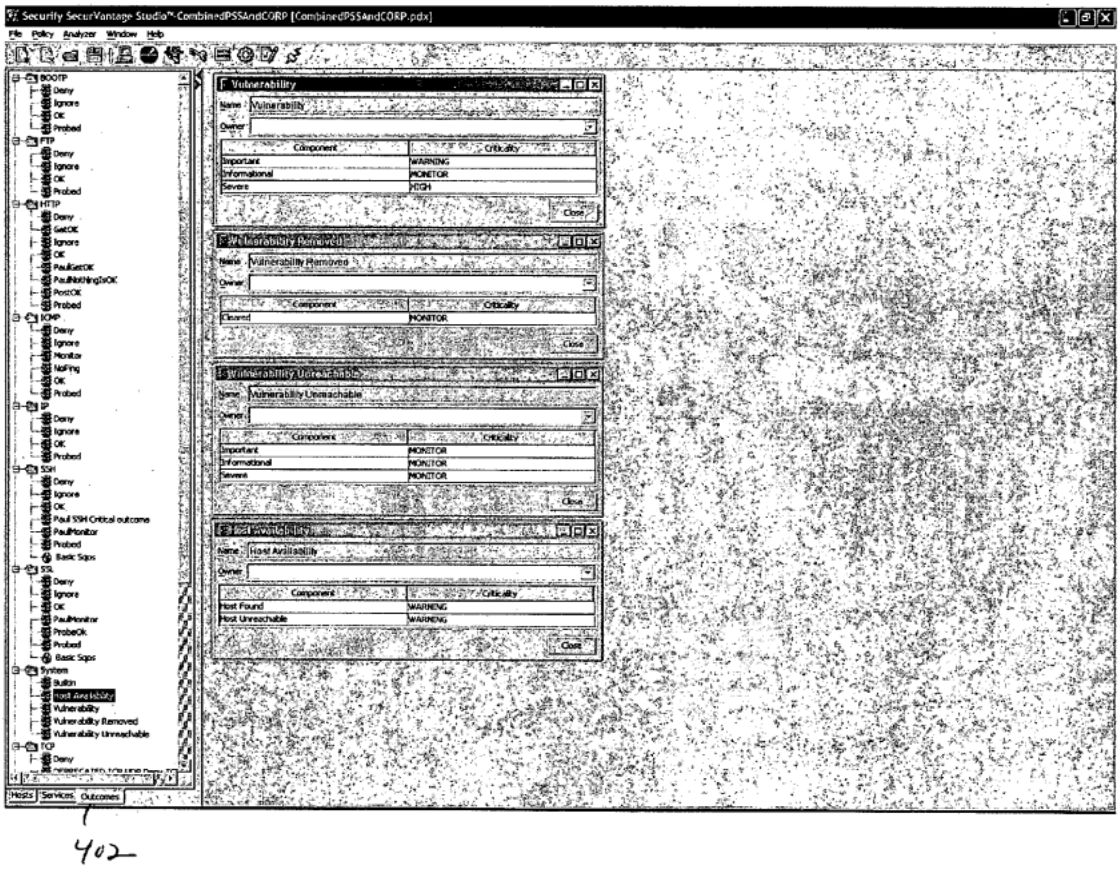
(Ex. 1006 at Figure 4).

215. Cooper also gives another example view wherein the outcomes of events are shown but this time with vulnerability event information.

> FIG. 5 is an example view from the studio module according to the invention. The outcomes tab 402 is selected in the panel window on the left, showing possible outcomes of events. Possible emitted events for vulnerabilities are shown along with the respective components and criticalities.
>
> (Ex. 1006 at 16:31-35).

216. See also, Cooper Figure 5, which I have reproduced hereinbelow:

Figure 5



(Ex. 1006 at Figure 5).

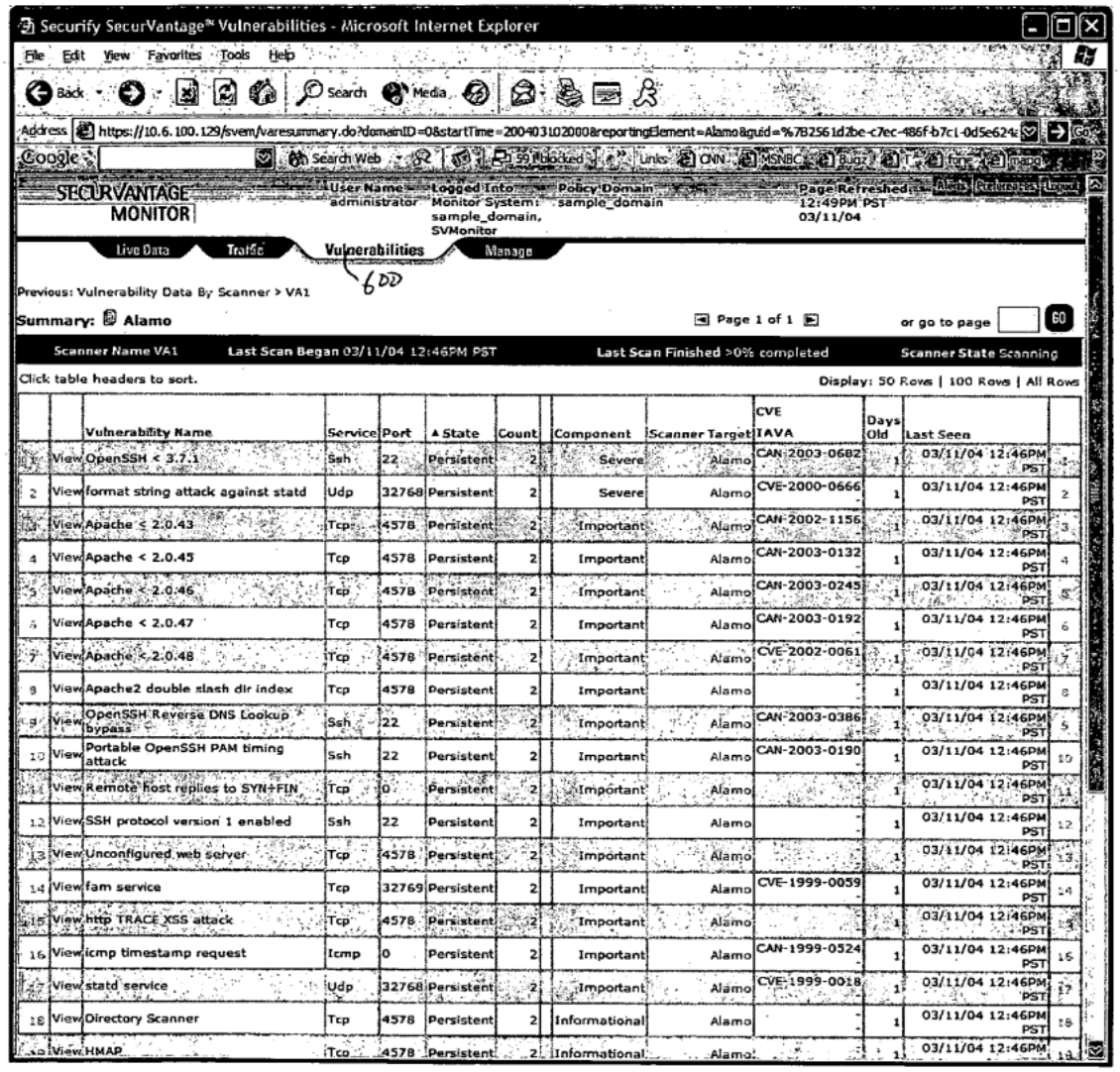
217. In yet another example, Cooper explains that the data can be displayed in a web page from a monitoring system. This view shows vulnerabilities in order of criticality and the temporal state.

> FIG. 6 is an example view of a Web page from the policy-based monitor system user interface according to the invention. Specifically, a page from the SecurVantage™ Monitor system is shown. The Vulnerabilities tab 600 is selected, whereby a complete list of vulnerabilities is shown in order of the criticality component, severe, important, and informational. It should be appreciated that among other information, each persistent state was seen twice (Count=2). It should also be appreciated that relevant CVE and IAVA data is displayed.

(Ex. 1006 at 16:36-44).

218. See also, Cooper Figure 6, which I have reproduced hereinbelow:

Figure 6



		Vulnerability Name	Service	Port	State	Count	Component	Scanner Target	CVE TAVA	Days Old	Last Seen	
1	View	OpenSSH < 3.7.1	Ssh	22	Persistent	2	Severe	Alamo	CAN-2003-0682	1	03/11/04 12:46PM PST	1
2	View	format string attack against statd	Udp	32768	Persistent	2	Severe	Alamo	CVE-2000-0666	1	03/11/04 12:46PM PST	2
3	View	Apache < 2.0.43	Tcp	4578	Persistent	2	Important	Alamo	CAN-2002-1156	1	03/11/04 12:46PM PST	3
4	View	Apache < 2.0.45	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0132	1	03/11/04 12:46PM PST	4
5	View	Apache < 2.0.46	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0245	1	03/11/04 12:46PM PST	5
6	View	Apache < 2.0.47	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0192	1	03/11/04 12:46PM PST	6
7	View	Apache < 2.0.48	Tcp	4578	Persistent	2	Important	Alamo	CVE-2002-0061	1	03/11/04 12:46PM PST	7
8	View	Apache2 double slash dir index	Tcp	4578	Persistent	2	Important	Alamo	-	1	03/11/04 12:46PM PST	8
9	View	OpenSSH Reverse DNS Lookup bypass	Ssh	22	Persistent	2	Important	Alamo	CAN-2003-0386	1	03/11/04 12:46PM PST	9
10	View	Portable OpenSSH PAM timing attack	Ssh	22	Persistent	2	Important	Alamo	CAN-2003-0190	1	03/11/04 12:46PM PST	10
11	View	Remote host replies to SYN+FIN	Tcp	0	Persistent	2	Important	Alamo	-	1	03/11/04 12:46PM PST	11
12	View	SSH protocol version 1 enabled	Ssh	22	Persistent	2	Important	Alamo	-	1	03/11/04 12:46PM PST	12
13	View	Unconfigured web server	Tcp	4578	Persistent	2	Important	Alamo	-	1	03/11/04 12:46PM PST	13
14	View	fam service	Tcp	32769	Persistent	2	Important	Alamo	CVE-1999-0059	1	03/11/04 12:46PM PST	14
15	View	http TRACE XSS attack	Tcp	4578	Persistent	2	Important	Alamo	-	1	03/11/04 12:46PM PST	15
16	View	icmp timestamp request	Icmp	0	Persistent	2	Important	Alamo	CAN-1999-0524	1	03/11/04 12:46PM PST	16
17	View	statd service	Udp	32768	Persistent	2	Important	Alamo	CVE-1999-0018	1	03/11/04 12:46PM PST	17
18	View	Directory Scanner	Tcp	4578	Persistent	2	Informational	Alamo	-	1	03/11/04 12:46PM PST	18
19	View	HMAP	Tcp	4578	Persistent	2	Informational	Alamo	-	1	03/11/04 12:46PM	19

(Ex. 1006 at Figure 6).

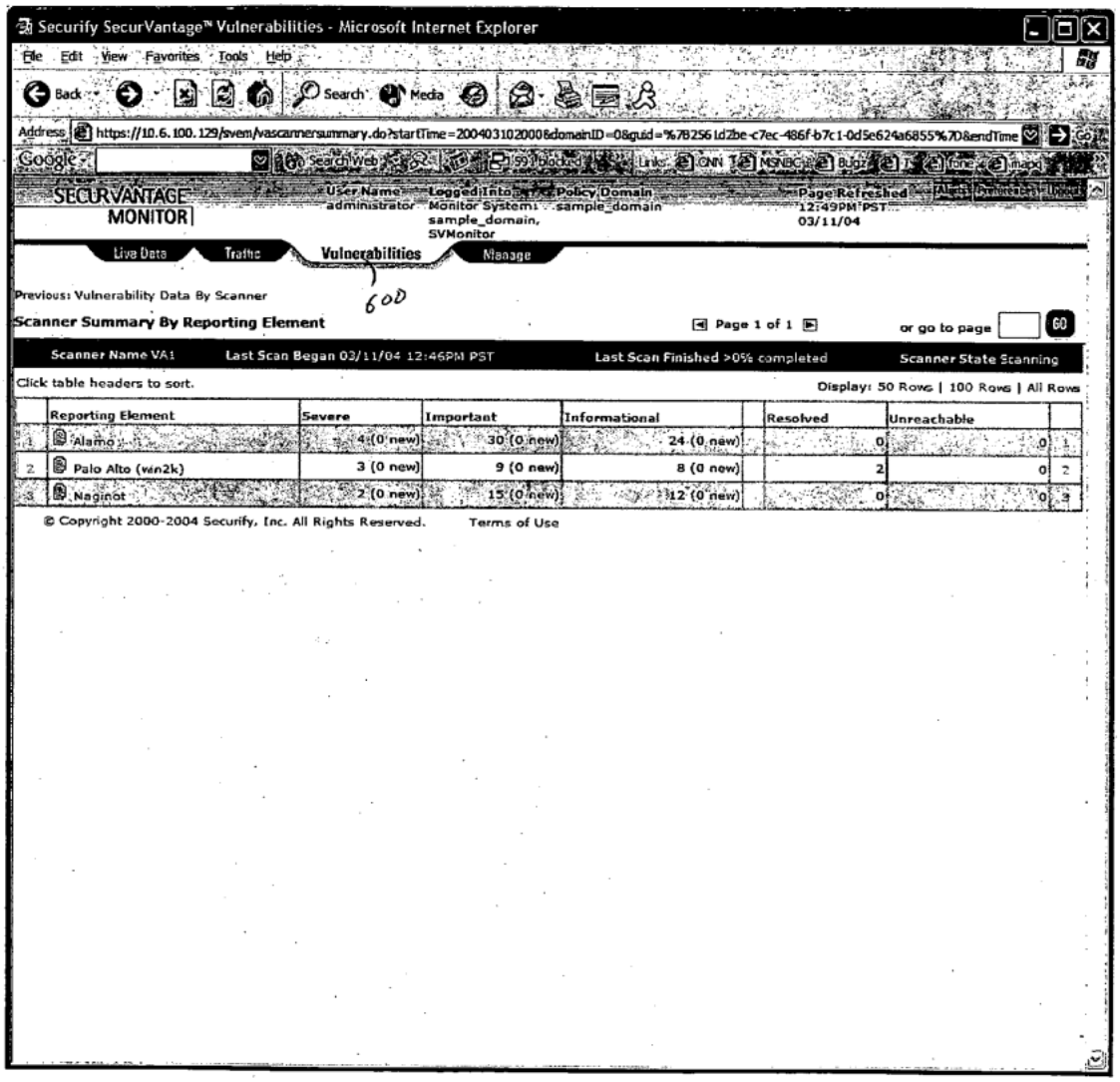
219. Cooper also describes another web-page based example that shows vulnerabilities along with a summary of the vulnerability scanner's findings.

> FIG. 7 is an example view of a Web page from the policy-based monitor system user interface according to the invention. The Vulnerabilities tab

600 is selected, whereby the specific view shown is Scanner Summary by Reporting Element, of which there are three in this example. (Ex. 1006 at 16:46-49).

220. See also, Cooper Figure 7, which I have reproduced hereinbelow:

Figure 7



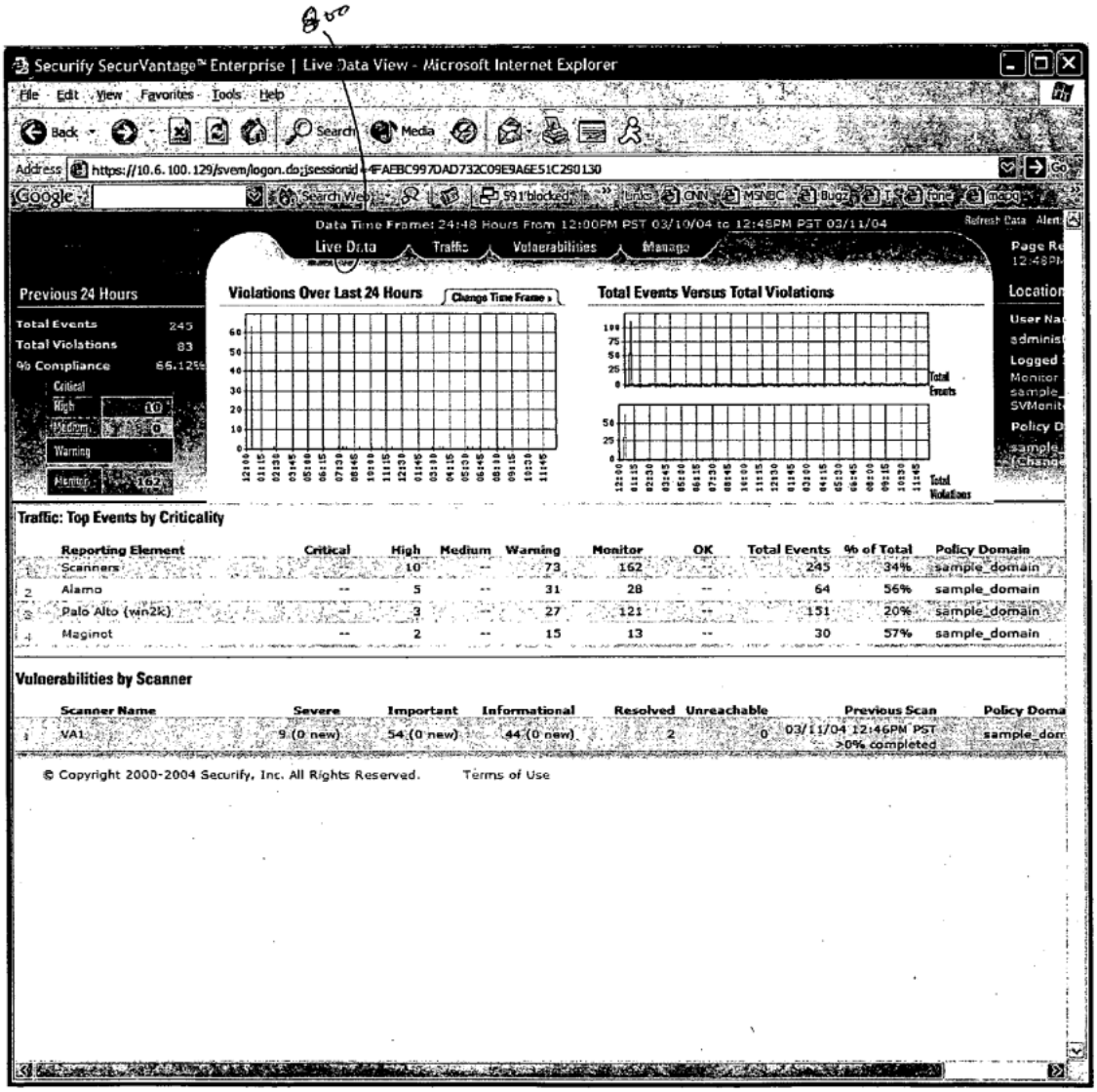
(Ex. 1006 at Figure 7).

221. In yet another example, Cooper describes another webpage that shows data related to events over the past 24 hours. This data includes the number of events and the number of events that violated the security policy.

> FIG. 8 is an example view of a Web page from the policy-based monitor system user interface according to the invention. The Live Data tab 800 is selected, whereby data from the previous 24 hours is shown. It should be appreciated that among other information, total events, total violations, and percent compliance is shown. Graphs showing total events versus total violations make is easy for the viewer to detect by visual means the vulnerability of the system at hand. Top events sorted by criticality is displayed in list form which makes it easy for the viewer to glean which reporting elements are impacted the most.
> (Ex. 1006 at 16:50-61).

222. See also, Cooper Figure 8, which I have reproduced hereinbelow:

Figure 8



(Ex. 1006 at Figure 8).

223. A POSITA would have recognized each of these examples as a "dashboard" of information and the collection of them as a "plurality of dashboards."

224. I have already explained with respect to limitations 1[a]-1[d] that Cooper identifies in real-time the operational integrity of applications. I incorporate this analysis into this section by reference. Thus, a POSITA would have understood these examples of Cooper to be providing

a status of such real-time assessment. Accordingly, the dashboards would also represent "realtime dashboards".

225. A POSITA would have found the teachings of Mukherjee and of Cooper, including with respect to limitation 1[e], to be compatible, because they both provide visibility into the health of the system for which an operator is responsible, and because they both display similar kinds of data. A POSITA would have understood the value of having multiple different views into such data, represented by multiple dashboards, for gaining insight into issues that may not be readily apparent by observing only a single metric.

226. Thus, I conclude that Mukherjee and Cooper each disclose limitation 1[e].

227. Thus, I conclude that Mukherjee in view of Cooper renders obvious Claim 1.

C. Claim 2 is rendered obvious by Mukherjee in view of Cooper

228. Claim 2 reads:

> *The method of claim 1, wherein the real-time operations of applications are monitored continuously.*

229. Mukherjee in view of Cooper renders obvious Claim 2.

230. Cooper discloses continuous scanning:

> The invention relates generally to the field of network security management. More particularly, the invention relates to ***scanning for vulnerabilities on a continuous basis*** and interpreting the resulting traffic in the context of policy.
> (Ex. 1006 at 1:21-24).

231. Furthermore, Cooper discloses that such continuous scanning serves to monitor "real-time operations of applications":

> The vulnerability assessment system presents current vulnerability state of the infrastructure. When continuous scanning is used, the vulnerability assessment system provides ***changes to this state based on new vulnerabilities found or existing vulnerabilities removed.*** Throughout the

> process the policy adds context ***in terms specific to the customer applications*** allowing prioritized action to be taken to remediate problems. (Ex. 1006 at 3:6-13).

232. I discussed in Section IX.A. why a POSITA would have been motivated to combine Mukherjee with Cooper in general. I also discussed with respect to limitation 1[d] that a POSITA would have been specifically motivated to combine Cooper's "continuous scanning" with Mukherjee's analysis of temporal context of events. I incorporate both of those analyses here by reference. For example, Mukherjee explains that, with respect to temporal context, "behavior which is unremarkable during standard working hours may be suspicious during off-hours." (Ex. 1005 at p. 13). A POSITA would have found Cooper's technique to be compatible with that of Mukherjee and would have expected a high likelihood of success in such a combination.

233. Thus, I conclude that Mukherjee in view of Cooper renders obvious Claim 2.

D. Claim 3 is rendered obvious by Mukherjee in view of Cooper

234. Claim 3 of the '948 patent, which depends from Claim 1, is rendered obvious by Mukherjee in view of Cooper as described below. Claim 3 reads:

> *The method of claim 1, wherein the event and behavior correlation engine receives one or more of: system events and infection profiles from a network activity sensor; integrity measurement and verification reports of scans from a system configuration sensor; system events pertaining to processor, network, memory, and storage resource consumption from a resource utilization sensor; and system events pertaining to image profiles and local execution context from an application integrity sensor.*

235. Mukherjee in view of Cooper renders obvious Claim 3.

236. Mukherjee discloses an "event and behavior correlation engine [that] receives one or more of…" In particular, the "DIDS director" contains a component called a "communications manager":

> The communications manager is responsible for the transfer of data between the director and each of the host and the LAN monitors. ***It accepts the notable event records from each of the host and LAN monitors*** and sends them to the expert system.
> (Ex. 1005 at p. 12, emphasis added).

237. As will be discussed hereinbelow, Mukherjee's DIDS director (or the communications manager thereof) receives data from sensors, including data a POSITA would have recognized as "event" and "behavior" data.

238. Mukherjee discloses "wherein the event and behavior correlation engine receives one or more of: ***system events and infection profiles from a network activity sensor.***"

239. Mukherjee teaches a LAN monitor capable of monitoring network activity.

> Like the host monitor, the LAN monitor consists of a LAN event generator (LEG) and a LAN agent. The LEG is currently a subset of the NSM [17, 27, 29]. Its main responsibility is to observe all of the traffic on its segment of the LAN in order to monitor host-to-host connections, services used, and volume of traffic. The LAN monitor reports on network activities such as rlogin and telnet connections, the use of security-related services, and changes in network traffic patterns.
> (Ex. 1005 at p. 12).

240. The LAN monitor then transmits these events, as described in my analysis of limitation 3[a]:

> The communications manager is responsible for the transfer of data between the director and each of the host and the LAN monitors. ***It accepts the notable event records from each of the host and LAN monitors*** and sends them to the expert system.
> (Ex. 1005 at p. 12, emphasis added).

241. The '948 patent specification does not give an explicit definition of a "network activity sensor" other than describing the data flow recited by Claim 3: "The event and behavior

correlation engine 130 is ***configured to receive network events and infection profiles 112 from a network activity sensor 111***…" (Ex. 1001 at 8:23-25, emphasis added).

242. Moreover, the specification does not give an explicit definition of "infection profiles 112". A similar term is used later, but it refers to data transmitted by a separate component ("network activity correlator 831"), which performs a different function than a "network activity sensor 111":

> In certain exemplary embodiments, the trust supervisor 848 receives an ***infection profile 832 for a network endpoint from a network activity correlator 831***, …
> (Ex. 1001 at 18:21-23, emphasis added).

> In embodiments, the ***infection profile 832*** comprises at least the victim endpoint address and one or more of a forensic confidence of the declared infection, the attacker source, the command and control source, the full evidence chain, and the infection time range (onset, duration).
> (Ex. 1001 at 18:48-52, emphasis added).

243. Because Mukherjee's LAN monitors "report on network activities" of various kinds by transmitting "event records" to the DIDS director (the claimed "event and behavior correlation engine"), a POSITA would have recognized that such a LAN monitor and its transmission teach the claimed "system events and infection profiles [received] from a network activity sensor."

244. Mukherjee further discloses "wherein the event and behavior correlation engine receives one or more of: … ***integrity measurement and verification reports of scans from a system configuration sensor***."

245. Mukherjee discloses a host monitor that collects information from an individual machine, including about "changes to the security state of the system". It also "report[s] anomalous behavior aggregated over time". In addition, as outlined hereinabove, the host agent transmits such information to the DIDS director (the claimed "behavior and event correlation engine"):

> The host monitor consists of a host event generator (HEG) and a host agent. The HEG collects and analyzes audit records from the host's operating system. ***The audit records are scanned for notable events***, which are transactions that are of interest independent of any other records. These include, among others, failed events, user authentications***, changes to the security state of the system***, and any network access such as rlogin and rsh. ***These notable events are then sent to the director for further analysis***. The HEG also ***tracks user sessions and report anomalous behavior aggregated over time*** through user and group profiles and the integration of the HAYSTACK intrusion detection algorithm [5] into DIDS. The host agent handles all communications between the host monitor and the DIDS director.
> (Ex. 1005 at pp. 11-12, emphasis added).

246. As I have discussed with respect to limitation 1[c], a POSITA would have understood data about "changes to the security state of the system" to represent data about the integrity of the system, as it may indicate impairments of components crucial to the secure operation of the system. A POSITA would also understand "anomalous behavior aggregated over time" to also represent data about the integrity of the system, as it may indicate unauthorized access that could impair the security or integrity of the system. Because Mukherjee's host monitor transmits these kinds of data, a POSITA would recognize that it teaches the claimed "integrity measurement and verification reports of scans [received] from a system configuration sensor."

247. Thus, I conclude that Mukherjee in view of Cooper renders obvious Claim 3.

E. Claim 4 is rendered obvious by Mukherjee in view of Cooper

248. Claim 4 of the '948 patent is rendered obvious by Mukherjee in view of Cooper. This claim, which depends from Claim 3, reads:

> *The method of claim 3, wherein the scans are performed as per a schedule, and wherein the schedule is one or more of hourly, daily, weekly, bi-weekly, or monthly, by a third party collaboration service to score the severity of an application based on the vulnerability, compliance, and patch status to affirm positive presence of a threat.*

249. Mukherjee in view of Cooper renders obvious Claim 4.

250. Mukherjee and Cooper both disclose "***wherein the scans are performed as per a schedule, and wherein the schedule is one or more of hourly, daily, weekly, bi-weekly, or monthly.***"

251. Mukherjee discloses that its monitoring can run daily:

> IDES monitors target system activity as it is recorded in audit records generated by the target system. Due to the fact that ***these profiles are updated daily***, IDES is able to adaptively learn a subject's behavior patterns; as users alter their behavior, the profiles change to reflect the most recent activity.
> (Ex. 1005 at p. 5).

252. Cooper describes that its system is capable of running on various schedules, including several of those recited in Claim 4:

> Using the policy-based enterprise manager system user interface the user configures ***how often a network scan is to take place, e.g. daily or weekly,*** and the maximum network bandwidth that it should utilize. This allows a network operator to schedule scans during off-peak periods and to limit network bandwidth utilization to an acceptable level.
> (Ex. 1006 at 8:55-60, emphasis added).
>
> The schedule information specifies when scans are to take place and is obtained from the monitor configuration file. Vulnerability scans are recurring events and the schedule information merely indicates the periodicity of the event. Following are examples of periodicity that may be specified:
>
> Continuous;
> ***Daily;***
> ***Every n days;***
> ***Day of the week (weekly); and***
> ***Day of the month (monthly).***
>
> (Ex. 1006 at 11:60-67, emphasis added).

253. Thus, Mukherjee and Cooper both disclose "wherein the scans are performed as per a schedule, and wherein the schedule is one or more of hourly, daily, weekly, bi-weekly, or monthly".

254. Mukherjee in view of Cooper further discloses "wherein the scans are performed… ***by a third party collaboration service to score the severity of an application based on the vulnerability, compliance, and patch status to affirm positive presence of a threat***."

255. Mukherjee discloses a security monitor that works with its IDES system: "The Information Security Officer's Assistant (ISOA) is a real-time security monitor implemented on a UNIX-based workstation that supports automated as well as interactive audit trail analysis [12, 13]." (Ex. 1005 at p. 5). ISOA is capable of examining "concern levels" based on the events received (e.g., events about an application):

> ISOAs analysis capabilities include both statistical as well as expert system components. These components cooperate in the automated examination of various "concern levels" of data analysis. ***As recognized indicators (sets of indicators) are matched, concern levels increase and the system begins to analyze increasingly detailed classes of audit events for the user or host in question.***
> (Ex. 1005 at p. 6, emphasis added).

256. A POSITA would have recognized that "concern levels" indicate higher or lower levels of potentially problematic (e.g., threatening) behavior being analyzed by the system, and would have recognized that such "concern levels" teach a concept similar to the claimed "scor[ing] the severity of an application" relative to its potential threat.

257. Cooper also discloses obtaining severity information about vulnerabilities, for example from a third-party tool called Nessus:

> Vulnerability Scanning technology is exemplified by commercial products such as the various scanners by Internet Security Systems, Inc. (ISS), the Cisco Scanner by Cisco Systems, Inc., and the ***Nessus Network Security Scanner***.
> (Ex. 1006 at 1:55-58, emphasis added).
>
> For each of the states hereinabove, a severity can be selected from the scanner, such as Nessus. For example, ***Nessus severity selections are: severe, informational, and important***.
> (Ex. 1006 at 10:12-15, emphasis added).

> ***The severity of the vulnerability, as defined by Nessus***, is mapped into one of the outcome components defined above. The mapping is as follows:
> Nessus output level HIGH is mapped to Severe;
> Nessus output level WARNING is mapped to Important; and
> Nessus output level NOTE is mapped to Informational.
> (Ex. 1006 at 12:61-67, emphasis added).

258. A POSITA would have recognized that the "severity selections" disclosed by Cooper ("severe, informational, important") indicate higher or lower levels of potentially problematic (e.g., threatening) behavior being analyzed by the system, and would have recognized that such "severity selections" teach a concept similar to the claimed "scor[ing] the severity of the application".

259. Furthermore, Cooper discloses that information it collects about vulnerabilities includes the following: "Common Identifier, a string containing one or more identifiers from common vulnerability repositories such as the Common Vulnerabilities and Exposures (CVE) number". (Ex. 1006 at 13:14-16). A POSITA would have understood that such CVE numbers identify that a recognized, documented, published vulnerability, known to practitioners of the art, and that if a vulnerability in a system has been identified as matching a particular CVE, that such identification "affirm[s] positive presence of a threat."

260. Cooper also discloses that its technique of detecting vulnerabilities includes detecting what a POSITA would recognize as teaching "compliance and patch status". For example, as I discussed in Section V.B., particular versions of software may be known to have particular vulnerabilities; when scanning for such vulnerabilities, scan results of Cooper include information about the software version:

> One embodiment of the invention adds vulnerability scanning capabilities to a policy-based monitor system and method to detect network services which contravene policy ***and to detect application-level vulnerabilities, e.g.***

> ***a host running a vulnerable version of an open source http server, such as Apache.***
> (Ex. 1006 at 4:10-15).

261. In other words, when Cooper detects a vulnerability related to a "vulnerable version" of some software (i.e., a version that does not include the patch or fix for such vulnerability), it includes information not only about the patch status but also about the compliance of such software to the policy of Cooper's "policy-based monitor system."

262. As I explained in Section IX.A., a POSITA would have been motivated to combine Mukherjee and Cooper generally. I incorporate that analysis here by reference. With reference to Claim 4, a POSITA would have recognized these teachings of Mukherjee and Cooper to be compatible, and as such would have expected a high likelihood of success in combining them.

263. Thus, Mukherjee in view of Cooper renders obvious Claim 4.

F. Claim 5 is rendered obvious by Mukherjee in view of Cooper

264. Claim 5 of the '948 patent, which depends from Claim 1, is rendered obvious by Mukherjee in view of Cooper as described below.

265. Claim 5 reads:

> *The method of claim 1, wherein the plurality of runtime dashboards are a plurality of application runtime reputation score dashboards that comprise one or more of: a global view dashboard; a system configuration dashboard; a resource utilization dashboard; an application integrity dashboard; and a network activity dashboard.*

266. As I explained with reference to limitation 1[e], Mukherjee and Cooper each disclose a "plurality of runtime dashboards". I incorporate that analysis here by reference. Mukherjee in view of Cooper further renders obvious the limitations of Claim 5.

267. Specifically, Mukherjee explains that the DIDS director includes a user-interface for permitting the System Security Officer (SSO) views into and control over the IDS.

> The director's user interface allows the SSO interactive access to the entire system. ***The SSO can watch activities on each host, watch network traffic*** (by setting "wiretaps"), and request more specific types of information from the monitors.
> (Ex. 1005 at p. 12, emphasis added).

268. In particular, one view exposed to the SSO is a numeric value representing the security state of the network:

> At the highest level, ***the model produces a numeric value between one and 100 which represents the overall security state of the network***. The higher the number, the less secure is the network. This value is a function of all the threats for all the subjects on the system. Here again we treat the collection of hosts as a single distributed system. Although representing the security level of the system as a single value seems to imply some loss of information, ***it provides a quick reference point for the SSO***.
> (Ex. 1005 at p. 13, emphasis added).

269. A POSITA would have understood such a score to be similar to the "reputation score" taught by the '948 patent.

270. A POSITA would have recognized that "network traffic" is a form of the recited "network activity". Because it is specifically taught as part of "[t]he director's user interface", a POSITA would have recognized that Mukherjee teaches a dashboard substantially similar to the claimed "network activity dashboard".

271. Moreover, Mukherjee anticipates combining its disclosures with additional components or tools:

> ***It is anticipated that a growing set of tools, including incident-handling tools and network-management tools, will be used in conjunction with the intrusion-detection functions of DIDS.*** This will give the SSO the ability to actively respond to attacks against the system in real time. Incident-handling tools may consist of possible courses of action to take against an attacker, such as cutting off network access, a directed investigation of a

> particular user, removal of system access, etc. Network-management tools that are able to perform network mapping would also be useful.
> (Ex. 1005 at p. 12, emphasis added).

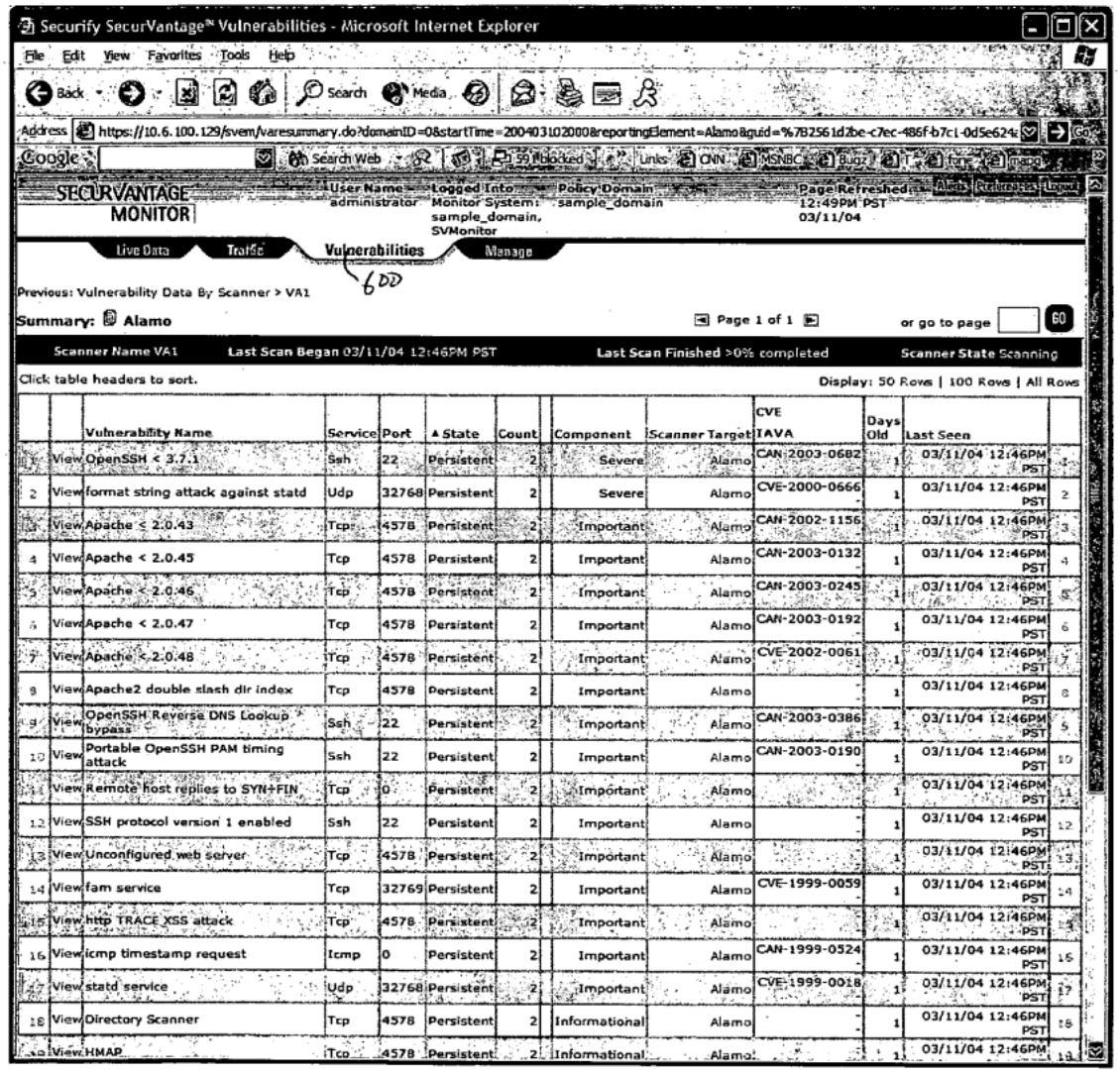
272. Because Mukherjee's stated intention for such extensibility with additional tools is to "give the SSO the ability to actively respond to attacks against the system in real time", and because Mukherjee's disclosed dashboard is a tool explicitly designed for the SSO to this end, a POSITA would have found it obvious that such a dashboard could be augmented or implemented with teachings from other references and would have been motivated to seek such additional references.

273. Cooper discloses several examples of dashboards. One example displays information about all vulnerabilities in the monitored network:

> FIG. 6 is an example view of a Web page from the policy-based monitor system user interface according to the invention. Specifically, a page from the SecurVantage™ Monitor system is shown. The Vulnerabilities tab 600 is selected, whereby ***a complete list of vulnerabilities is shown in order of the criticality component***, severe, important, and informational. It should be appreciated that among other information, each persistent state was seen twice (Count=2). It should also be appreciated that relevant CVE and IAVA data is displayed.
> (Ex. 1006 at 16:36-44, emphasis added).

274. See also, Cooper Figure 6, which I have reproduced hereinbelow:

Figure 6



		Vulnerability Name	Service	Port	State	Count	Component	Scanner Target	CVE TAVA	Days Old	Last Seen	
1	View	OpenSSH < 3.7.1	Ssh	22	Persistent	2	Severe	Alamo	CAN-2003-0682	1	03/11/04 12:46PM PST	1
2	View	format string attack against statd	Udp	32768	Persistent	2	Severe	Alamo	CVE-2000-0666	1	03/11/04 12:46PM PST	2
3	View	Apache < 2.0.43	Tcp	4578	Persistent	2	Important	Alamo	CAN-2002-1156	1	03/11/04 12:46PM PST	3
4	View	Apache < 2.0.45	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0132	1	03/11/04 12:46PM PST	4
5	View	Apache < 2.0.46	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0245	1	03/11/04 12:46PM PST	5
6	View	Apache < 2.0.47	Tcp	4578	Persistent	2	Important	Alamo	CAN-2003-0192	1	03/11/04 12:46PM PST	6
7	View	Apache < 2.0.48	Tcp	4578	Persistent	2	Important	Alamo	CVE-2002-0061	1	03/11/04 12:46PM PST	7
8	View	Apache2 double slash dir index	Tcp	4578	Persistent	2	Important	Alamo		1	03/11/04 12:46PM PST	8
9	View	OpenSSH Reverse DNS Lookup bypass	Ssh	22	Persistent	2	Important	Alamo	CAN-2003-0386	1	03/11/04 12:46PM PST	9
10	View	Portable OpenSSH PAM timing attack	Ssh	22	Persistent	2	Important	Alamo	CAN-2003-0190	1	03/11/04 12:46PM PST	10
11	View	Remote host replies to SYN+FIN	Tcp	0	Persistent	2	Important	Alamo		1	03/11/04 12:46PM PST	11
12	View	SSH protocol version 1 enabled	Ssh	22	Persistent	2	Important	Alamo		1	03/11/04 12:46PM PST	12
13	View	Unconfigured web server	Tcp	4578	Persistent	2	Important	Alamo		1	03/11/04 12:46PM PST	13
14	View	fam service	Tcp	32769	Persistent	2	Important	Alamo	CVE-1999-0059	1	03/11/04 12:46PM PST	14
15	View	http TRACE XSS attack	Tcp	4578	Persistent	2	Important	Alamo		1	03/11/04 12:46PM PST	15
16	View	icmp timestamp request	Icmp	0	Persistent	2	Important	Alamo	CAN-1999-0524	1	03/11/04 12:46PM PST	16
17	View	statd service	Udp	32768	Persistent	2	Important	Alamo	CVE-1999-0018	1	03/11/04 12:46PM PST	17
18	View	Directory Scanner	Tcp	4578	Persistent	2	Informational	Alamo		1	03/11/04 12:46PM PST	18
19	View	HMAP	Tcp	4578	Persistent	2	Informational	Alamo		1	03/11/04 12:46PM	19

(Ex. 1006 at Figure 6).

275. As I explained with reference to Claim 4, Cooper teaches "scor[ing] the severity of an application based on the vulnerability, compliance, and patch status to affirm positive presence of a threat." Claim 5 recites an application score in different words: "application runtime reputation score." The '948 patent specification does not define this term. However, it does mention "an application score 2406" which may be one component of a "reputation score":

> Referring to FIG. 24, the exemplary hierarchical representation of a subject reputation score 2400 includes a reputation score 2401, a transaction score 2409, a user score 2407, a service score 2408, a social score 2402, a professional score 2403, a device score 2404, an organization score 2405, and ***an application score 2406***. In the exemplary embodiment of FIG. 24, the organization score 2405 can be, for example a company score 2405.
>
> A reputation score may comprise ***one or more component scores*** (as an aggregation), where a component score is illustrated in FIG. 24 below an aggregate score (for example, a service score may be a component score of a transaction score). An aggregate reputation score may provide an enumerated list to annotate component scores included.
> (Ex. 1001 at 35:13-26).

276. It is my opinion that the "score [of] the severity of an application" recited in Claim 4 and the "application runtime reputation score" recited in Claim 5 represent the same or substantially similar concepts, and that a POSITA would have understood these terms as such. In Cooper's dashboard cited above, vulnerabilities are "shown in order of the criticality component, severe, important, and informational", that is, the "criticality component" is displayed and is also used for sorting. Because the "criticality component" teaches a concept substantially similar to the "application runtime reputation score" and/or the "score [of] the severity of an application" (*see* my analysis of Claim 4), and because Cooper's dashboard incorporates such "criticality component" (an "application runtime reputation score"), a POSITA would have understood Cooper's dashboard to teach one of the claimed "plurality of application runtime reputation score dashboards".

277. An example of a "global view dashboard" given in the '948 patent specification, which is given in the context of "an exemplary GUI for a data center administration console" (i.e., the data center encompasses all the machines being monitored by the system), displays information relating to "the data center inventory" (i.e., the machines in the data center). It states:

> FIG. 14 depicts an exemplary GUI for a data center administration console 1400 including an operational integrity dashboard 1450.

> Referring to FIG. 14, the exemplary data center administration console 1400 may visually display as a single pane an operational integrity dashboard or display window 1450 to provide a 'global view' of the data center inventory 1401 (i.e., a 'global view dashboard'). The display window 1450 can include information fields…
> (Ex. 1001 at 27:59-67).
>
> Although the system configuration dashboard 1550 is shown with specific information fields and labels in a particular arrangement, it is contemplated that other information fields and arrangements are possible.
> (Ex. 1001 at 28:52-55).

278. Cooper teaches displaying information from all the machines being monitored, namely "a complete list of vulnerabilities" found on the monitored machines. Because Cooper describes the scope of its displayed information in similar terms to the terms used in the '948 patent specification, a POSITA would have understood Cooper's disclosures to be substantially similar. Furthermore, because the specification states that "it is contemplated that other information fields and arrangements are possible," and because Claim 5 does not specifically recite information fields that should be present in a "global view dashboard," a POSITA would have found it obvious to include such vulnerability information as Cooper provides when implementing a "global view dashboard."

279. As I explained in Section IX.A., a POSITA would have been motivated to combine the teachings of Mukherjee with those of Cooper. Relative to Claim 5, a POSITA would have observed that Mukherjee teaches displaying information in a dashboard, including certain types of information to display, but does not give much detail on how to implement such a dashboard. A POSITA would have been motivated to look to a reference like Cooper for examples on how to implement such a dashboard. Furthermore, a POSITA would have found the integration to be straightforward and have a high likelihood of success, because Mukherjee anticipates the need and deals with information similar to that collected and presented by Cooper's dashboards.

280. Thus, I conclude that Mukherjee in view of Cooper renders obvious Claim 5.

X. THERE ARE NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

281. I am not aware of any objective indicia of non-obviousness related specifically to the ’948 patent. Indeed, I am not aware of any evidence of commercial success, long felt but unsolved need, failure of others, industry praise, or unexpected results specifically related to the ’948 patent.

XI. CONCLUSION

282. I declare that all statements made herein are true to the best of my knowledge and that all statements made on information and belief are believed to be true, and that this Declaration is made with 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.

Executed on 6 April, 2023

By: _____
Seth James Nielson