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.

_____

IPR2023-00799

U.S. Patent No. 9,092,616

DECLARATION OF SETH JAMES NIELSON

TABLE OF CONTENTS

I. INTRODUCTION 1

II. QUALIFICATIONS 2

III. MATERIALS CONSIDERED 12

IV. LEGAL STANDARDS 14

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

A. Intrusion Detection Systems 17

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

C. Code Evasion, Obfuscation, and Packing 18

VI. OVERVIEW OF THE CHALLENGED PATENT 20

A. THE ’616 PATENT SPECIFICATION AND CLAIMS 20

B. PROSECUTION HISTORY 21

VII. LEVEL OF SKILL IN THE ART 23

VIII. Overview of the Prior art 23

A. OVERVIEW OF “NETWORK INTRUSION DETECTION” TO MUKHERJEE (“MUKHERJEE”) 23

B. OVERVIEW OF US20070192867 TO MILIEFSKY 26

C. OVERVIEW OF US PATENT 8,104,089 TO GUO 32

IX. GROUND 1: Claims 1-6 are rendered obvious by MUKHERJEE IN VIEW OF MILIEFSKY 35

A. Motivation to Combine Mukherjee and Miliefsky 35

B. Claim 1 is rendered obvious by Mukherjee in view of Miliefsky 37

1. Limitation 1[a] 37

2. Limitation 1[b] 49

3. Limitation 1[c] 51

4. Limitation 1[d] 51

5. Limitation 1[e] 52

6. Limitation 1[f] 58

7. Limitation 1[g] 61

8. Limitation 1[h] 64

C. Claim 2 is rendered obvious by Mukherjee in view of Miliefsky 66

D. Claim 3 is rendered obvious by Mukherjee in view of Miliefsky 67

E. Claim 4 is rendered obvious by Mukherjee in view of Miliefsky 69

F. Claim 5 is rendered obvious by Mukherjee in view of Miliefsky 70

G. Claim 6 is rendered obvious by Mukherjee in view of Miliefsky 72

1. Limitation 6[a] 72

2. Limitation 6[b] 72

X. Ground 2: Claim 7 is rendered obvious by Mukherjee in view of Miliefsky in further view of Guo 75

A. Motivation to Combine Mukherjee and Miliefsky and Guo 75

B. Claim 7 is rendered obvious by Mukherjee in view of Miliefsky in further view of Guo 77

XI. There are no objective indicia of nonobviousness 81

XII. Conclusion 82

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. 9,092,616 ("the '616 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 '616 patent are obvious in light of the prior art discussed below. The '616 patent is directed to general application of known network security analysis techniques to the attestation of system operational integrity.

4. In summary, my conclusion is that the claims of the '616 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 1-6 are rendered obvious to a POSITA by the combination of "Network Intrusion Detection" by Mukherjee et al. ("Mukherjee") in view of U.S. Patent Application 2007/0192867 to Miliefsky ("Miliefsky").

b. Claim 7 is rendered obvious to a POSITA by the combination of "Network Intrusion Detection" by Mukherjee et al. ("Mukherjee") in view of U.S. Patent Application 2007/0192867 to Miliefsky ("Miliefsky") in further view of U.S. Patent No. 8,104,089 to Guo et al. ("Guo").

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 '616 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. 9,092,616 to Kumar et al. ("the '948 patent")
- Ex. 1002: Prosecution history of the '616 patent
- Ex. 1003: Expert Declaration of Seth James Nielson, Ph.D.
- Ex. 1004: Curriculum Vitae of Seth James Nielson, Ph.D.
- Ex. 1005: Biswanath Mukherjee, Network Intrusion Detection, IEEE Network, (June 1994). ("Mukherjee")
- Ex. 1006: U.S. Published Patent Application No. 2007/0192867 to Miliefsky ("Miliefsky")
- 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 No. 8,104,089 to Guo et al. ("Guo")
- Ex. 1012: Aubrey-Derrick Schmidt, Static Analysis of Executables for Collaborative malware Detection on Android, IEEE Xplore, (2009).
- Ex. 1013: U.S. Patent Publication No. 2009/0328222 to Helman
- Ex. 1014: U.S. Patent Publication No. 2012/0072968 to Wysopal
- Ex. 1015: U.S. Patent Publication No. 2012/0110174 to Wootton
- Ex. 1016: J. Andrew Schafer, Windows Management Instrumentation: Create WMI Providers to Notify Applications of System Events, msdn magazine, (October, 2019).
- Ex. 1017: U.S. Patent Publication No. 2008/0244748 to Neystadt et al.
- Ex. 1018: Ilsun You, Malware Obfuscation Techniques: A Brief Survey, IEE Computer Society, 297-300, (2010).
- Ex. 1019: Mamoun Alazab, Towards Understanding Malware Behaviour by the Extraction of API calls, IEE Computer Society, 52-59, (July, 2010).
- Ex. 1020: Request for Comments (RFC) 5209, Network Endpoint Assessment (NEA): Overview and Requirements, (June, 2008).

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 '616 patent that would have been known to a POSITA. 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). (*See* Ex. 1006 at [0058]). 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. Code Evasion, Obfuscation, and Packing

69. Because a many approaches to detecting malware rely, in one form or another, on comparing potential malware to a known signature, malware authors employ various techniques

to hide their malware from signature-based detection systems. These approaches often take the form of "code evasion," wherein the malicious code operates in a way to evade a detection system.

70. One common approach to evasion is "obfuscation." Code obfuscation is "a technique that makes programs harder to understand… [and to do so] it converts a program to a new different version while making them functionally equal to each other." (Ex. 1018 at p. 1). Malware commonly achieves obfuscation by the use of "packers".

> Packers are commonly used today for code obfuscation or compression. Packers are software programs that are able to be used to compress and encrypt the [executable] in secondary memory and restore the original executable image, when loaded into main memory (RAM). Recently, malware authors have used packers to avoid detection.... Packing the malware makes the obfuscation method difficult to understand and the malware authors only need to change a small number of lines of code in order to change the malware signature.
> (Ex. 1019 at p. 4).

71. Malware authors can also combine packers. As explain by Guo, one of the prior art references used in Ground 2 hereinbelow, "[m]alware writers… can apply packing multiple times to a malware O and generate such variants as P1(P2(O)), P2(P1(O)), and so on. By applying multi-layer packing, even with a small number of packers, malware writers can easily generate a large number of variants from the same malware." (Ex. 1011 at 3:44-51).

72. However, even though code obfuscation is used by malware authors, it is also used for legitimate purposes such as protecting legitimate programs from unauthorized analysis. "Originally, this [obfuscation] technology aimed at protecting the intellectual property of software developers." (Ex. 1018 at p. 1). Accordingly, the existence of obfuscation is not sufficient to positively identify malware. Rather, the code must be unpacked and examined to determine if it is malicious.

VI. OVERVIEW OF THE CHALLENGED PATENT

A. THE ’616 PATENT SPECIFICATION AND CLAIMS

73. The ’616 patent claims disclose a method with the goal of “providing real-time operational integrity of a system.” (Ex. 1001 at Claim 1). The method accomplishes this with an “attestation service”, with inputs received from “a network trust agent, an endpoint trust agent, and a trust orchestration server.” (*Id.*). These agents collect information about events, actions, and applications, and the system uses that information, in conjunction with “third party network endpoint assessments” to generate “an integrity profile for the system.” (*Id.*). Dependent claims disclose various embodiments of this method and specify useful data sources and ways of analyzing the data. (*See id.* at Claims 2-7). Based on the usage of “third party” in claims 1 and 2, Claim 1’s use of third party” is broader than an “endpoint assessment service” and would include assessments produced not just by “services” but also by third parties in other ways, or merely other vendors. (Ex. 1001 at Claims 1 and 2).

74. Claim 7 of the ’616 patent refers to “reverse engineered code,” which patent describes as follows:

> At this point, the malware analyzer 540 reports capabilities detected (***present in reverse engineered code***) ***through static and dynamic analysis of the image***. Such image analysis can look for code obfuscation techniques ***in the reverse engineered code*** such as anti-debug, anti-trace, anti-memory, and anti-emulation, [etc.]
> (Ex. 1001 at 20:12-17, emphasis added).

75. The preceding paragraph states:

> Assuming no checks fail at this stage, the ***process P(benign) is continuously monitored by the integrity processor 630 for explicit rule violations***. In an embodiment, violations of a rule in a ruleset 631 are continuously monitored for and detected. These rules detect anomalous behaviors in benign processes. Once a rule trigger criteria is reached, the endpoint trust agent 410 initiates an image integrity check and leverages a remote malware analyzer 540 through the trust orchestrator 430. The purpose of this check is to avoid complete reliance on supply chain

> provenance and locally available evidence. ***The image profile 519 generated from malware binary analysis provides the integrity processor additional local events to monitor for process P(suspect)***.
> (Ex. 1001 at 19:66-20:11, emphasis added).

76. Thus, in this context, the reverse engineering seems to be undoing the obfuscation through static and dynamic analysis. Through static and dynamic analysis, the code obfuscation can be reversed (i.e., "reverse engineered") in order to either identify the malware or at least identify that known obfuscation techniques are in use (e.g., such as "AntiDbg" as illustrated in Figure 7A).

77. While the '616 patent specification is directed to attestation of application security and user reputation within the field of data center virtualization, the particular claims of the '616 patent are directed more narrowly to "providing an attestation service for providing runtime operational integrity of a system." (Ex. 1001 at 1:15-18, Claim 1).

B. PROSECUTION HISTORY

78. I understand that prosecution of the '616 patent began with a Non-final Office Action on July 18, 2013 that rejected claims 1-14, and 17-20. I understand claims 1-7 were rejected under 35 U.S.C. §102(b) as anticipated by Pat. Pub. 2009/0328222 to Helman. Furthermore, claims 8-10 were rejected under 35 U.S.C. §103(a) as unpatentable over Helman and USP 8,214,497 to Alperovitch. Similarly, claims 11-13 were rejected under 35 U.S.C. §103(a) as unpatentable over Helman-Alperovitch and Pat. Pub. 2012/0110174 to Wotton. Claim 14 was rejected under 35 U.S.C. §103(a) as unpatentable over Helman and Official Notice. Claim 17 was rejected under 35 U.S.C. §103(a) as unpatentable over Helman-Alperovitch and "Static Analysis of Executables for Collaborative malware Detection on Android" ("Schmidt") and Pat. Pub. 2012/0072968 to Wysopal. Furthermore, claims 18-19 were rejected under 35 U.S.C. §103(a) as unpatentable over Helman-Alperovitch-Schmidt-Wysopal and Official Notice. Finally, claim 20

was rejected under 35 U.S.C. §103(a) as unpatentable over Helman and "Windows Management Instrumentation: Create WMI Providers to Notify Applications of System Events" ("msdn"). (Ex. 1002 at pp. 276-304).

79. On October 1, 2013, I understand that Taasera responded by amending claims 1, 4-8, 11-17, and 20 and arguing differences between the claim language and Helman. (*Id*. at pp. 201-232).

80. I understand that in a Final Office Action dated January 17, 2014, the examiner rejected claims 1-7 and 14-16 under 35 U.S.C. §103(a) as unpatentable over Helman, Pat. Pub. 2008/0244748 to Neystadt, and Wootton; rejected claims 8-13 under 35 U.S.C. §103(a) as unpatentable over Helman, Alperovitch, and Wootton; rejected claim 17 under 35 U.S.C. §103(a) as unpatentable over Helman-Alperovitch-Wootton, and Schmidt; rejected claims 18 and 19 under 35 U.S.C. §103(a) as unpatentable over Helman-Alperovitch-Wooton-Schmidt-Wysopal and Neystadt; and rejected claim 20 under 35 U.S.C. §103(a) as unpatentable over Helman-msdn and Wootton. (*Id*. at pp. 135-172).

81. After an Examiner-Initiated Interview, Taasera responded by amending claim 1 to recite, in part "<u>sending, by the endpoint trust agent on a monitored device, a dynamic context including endpoint events and actions of the monitored device and applications executing on the monitored device at runtime</u>... <u>the</u> applications executing <u>on the monitored device at runtime</u>". Furthermore, Taasera amended claim 8 to recite in part "generating, by the endpoint trust agent <u>on an endpoint device</u>". (*Id*. at pp. 91-94; underlines in original). Taasera also argued for the novelty of claim 1 and submitted that the arguments were also applicable to independent claims 8 and 20. (*Id*., p.103). Taasera argued that the cited references did not disclose "sending, by the endpoint trust agent on a monitored device, a dynamic context including endpoint events and

actions of the monitored device and applications executing on the monitored device at runtime... the applications executing on the monitored device at runtime." (*Id*).

82. I understand that after this interview, a Notice of Allowance issued on May 22, 2015 with an Examiner's statement of reasons for allowance that "applicant's arguments … are persuasive, as such the reasons for allowance are in all probability evident from the record…." (*Id*. at pp. 33-39).

VII. LEVEL OF SKILL IN THE ART

83. 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 '616 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.

84. 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" TO MUKHERJEE ("MUKHERJEE")

85. "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).

86. In Mukherjee, "several host-based and network-based IDSs are surveyed, and the characteristics of the corresponding systems are identified." (*Id.*). Many of these systems build upon each other or utilize components of other systems. For example, one of the systems to which Mukherjee devotes particular attention, called DIDS and explained in further detail hereinbelow, builds on and uses components of NSM, a system developed by a different group. (*See id.* at pp. 8 and 11). As Mukherjee explains, "[DIDS] is an outgrowth of the NSM project. DIDS is designed to guard against some of NSM's deficiencies…" (*Id.* at p. 11). In particular, one component of DIDS, the "LAN event generator (LEG)… is currently a subset of the NSM…" (*Id.* at p. 12). Thus, while Mukherjee conducts a survey of many systems in this solution space, it is explicit that many components are combinable and work together to achieve their goals.

87. 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).

88. 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.*).

89. 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).

90. 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.*).

91. 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.*).

92. 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).

93. 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.*).

94. Mukherjee also discloses systems for displaying information to system administrators. One example is 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).

95. 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 US20070192867 TO MILIEFSKY

96. U.S. Patent Application Publication 2007/0192867 to Miliefsky ("Miliefsky") is directed toward "… computer security, and more particularly to a micro-appliance for use in

defending against common vulnerabilities and exploits." (Ex. 1006 at [0004]). With respect to the existing art, Miliefsky notes, "[t]he main problem with conventional anti-hacker security system is that they are not designed to stop hackers, instead they are countermeasures that react to threats." (*Id.* at [0007]). Additionally, "[a]nother problem with conventional anti hacker security systems is that they are typically built as proprietary systems, resulting in long design, development and release cycles." (*Id.* at [0008]).

97. To address these issues Miliefsky discloses automatically and proactively identifying and remediating the vulnerabilities used by attackers to exploit a system.

> The system and methods described herein include, among other things, security systems that provide ***proactive automated defense against hackers by automatically finding, reporting, communicating with countermeasures about and removing the common vulnerabilities and exposures (CVEs) that they exploit***. Accordingly, the systems described herein provide for proactive security by determining the components that exist on a network system and generating a list of network assets.
> (Ex. 1006 at [0039]).

98. As a first step, Miliefsky explains that the system may scan for existing devices to create an inventory. As part of this scan "rogue" devices (i.e., devices that are vulnerable) are identified.

> As part of the security method, in a first step the network is scanned and/or probed for any and all attached equipment and related assets, herein referred to as "network-based" assets. The method will dynamically detect and map changes to LAN and WAN connected equipment including searching for equipment which may be deemed as rogue and creating a network-based assets list, wherein the list contains information as to the location of the network-based assets.
> (*Id.* at [0040]).

99. Miliefsky's teachings also include pulling on public sources of vulnerability information, such as Common Vulnerabilities and Exposures (CVEs), to identify known problems in the system. The CVEs can also be provided to build a "true risk profile."

> In a further step, the method audits one or more of the network-based assets for common vulnerabilities and exposures (CVEs) as defined by the U.S. federally funded CVE list managed by MITRE corporation or any similar list. The method will generate a CVE and related regulatory compliance audit reports and update the CVE and related regulatory compliance audit tests. In a further step the method can share MAC, IP, Port, CVE and related regulatory compliance other related audit data with various INFOSEC countermeasures designed to help protect network-based assets against attacks…
>
> The method defines a true risk profile for the computer-based network environment, and uses the knowledge of external and internal CVEs as well as how to manage and remediate against these CVEs, to provide more robust and proactive security.
> (*Id.* at [0043]-[0045]).

100. A POSITA would have known that CVEs represent publicly available third-party vulnerability data. (*See* Section V.B.).

101. Miliefsky's disclosures include a component that utilizes such third-party vulnerability data to make security assessments of the network as a whole and of individual components on the network:

> An Information Security (INFOSEC) software engine acts as a gateway between users, personal computers, servers, services and the computer network (internet, intranet, extranet, wide area network, wireless network or local area network). The information Security (INFOSEC) Engine controls the computer-based network scanning, standards-based vulnerability assessment through common vulnerabilities and exposures (CVEs) testing, reporting and remediation as well as interfacing with the INFOSEC ENGINE PLUGINs. The INFOSEC Engine is structured in a modular fashion with a main controller that takes input for control from the Secure GUI modules. Functions include reading and acting upon the configuration and scheduling data as stored by the Secure GUI modules. The INFOSEC Engine contains a unique module for each vulnerability assessment CVE test as well as communication modules to enable non-intrusive testing for each unique IP Address accessible from the computer-based network. The INFOSEC Engine contains read, write and export functionality for vulnerabilities found and reported in various formats including but not limited to structured query language (SQL) databases and tables, portable document format (PDF), extensible markup language (XML), hypertext markup language (HTML), comma separated values (CSV) and Excel file format (XSL). The INFOSEC Engine, at initialization,

> is able to determine which CVE tests are available as well as which INFOSEC Engine Plugins are available and then to relay this information to the Secure GUI for administration, control and management.
> (Ex. 1006 at [0049]).

102. Information about these known vulnerabilities can be stored and transmitted in a standard format called OVAL, which can facilitate exchange of data with third-party systems:

> OVAL is the Open Vulnerability Assessment Language. It is funded by the U.S. Department of Homeland Security (DHS) and in summary the XML, machine readable format for the Common Vulnerabilities and Exposures (CVE®) standard. OVAL is an international, information security community baseline standard for how to check for the presence of vulnerabilities and configuration issues on computer systems. OVAL standardizes the three main steps of the process with an OVAL System Characteristics Schema for collecting configuration data from systems for testing; OVAL Definitions to test for the presence of specific vulnerabilities, configuration issues, and/or patches; and an OVAL Results Schema for reporting the results from the evaluated systems. The tests are standardized, machine-readable XML Vulnerability Definitions, Compliance Definitions, and Patch Definitions. OVAL's schemas and definitions are all free to download, use, reference, and implement. An "OVAL-compatible" tool, service, Web site, database, or advisory/alert uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state. An "OVAL-ID compatible" tool, Web site, database, archive, or security advisory includes OVAL-ids as part of the information it conveys about a security issue, and provides for searching by OVAL ID with potential linkage back to the source definition of the OVAL-ID.
> (Ex. 1006 at [0103]-[0104]).

103. Miliefsky teaches an architecture that can support many components working together. Part of this architecture is responsible for "clientless" functions, such as Network Admission Control (NAC). See, for example, Figure 11, reproduced hereinbelow:

Fig. 11

Simplified Vulnerability Management and Clientless Network Admission Control System:

a. Detect trusted and untrusted network assets, block and alert untrusted hosts or audit and block ports on trusted hosts with CVEs.

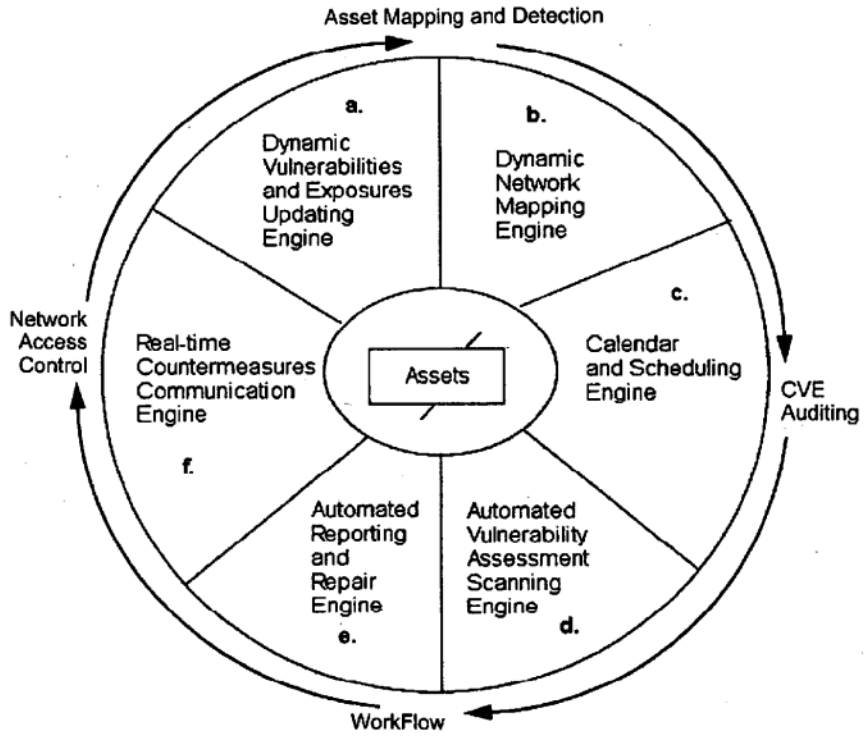
b. Map the local area network for trusted and untrusted IP Addresses with MAC Address and Operating System (OS) information.

c. Calendar and scheduling engine for scheduled auditing

d. Once scheduled, the engine performs vulnerability scans for CVEs in each asset

e. Reports are then generated and workflow process begins for the repair (manual or automatic) of the CVEs, which have been discovered.

f. The countermeasure communications engine performs the key functionality for clientless NAC by driving firewalls, VPNs and SmartSwitches to be automatically reconfigured through remote control using their published application programming interfaces (APIs) through secure means such as OPSEC or authenticated SSH and command line interfaces.



(Ex. 1006 at Figure 11).

104. Some of these components may run on "open box hardware", which Miliefsky defines as a wide range of standard or commonly-available hardware and software:

> Open-Box Hardware is defined as any computer-based system that can operate standards-based software and operating systems included but not limited to Linux, BSD, Unix or Windows on Intel, AMD or compatible hardware systems. The Structure of the Open-Box Hardware can range from hand-held wired or wireless computer equipment to standard portable

> digital assistants (PDAs), laptops, desktops, server or other computers. The functionality provided must include basis operating system, application and computer-based network connectivity.
> (Ex. 1006 at [0054]).

105. Components running on such hardware (e.g., the Countermeasures Communication Engine) are able to communicate with existing purpose-built "countermeasures" already deployed on the network (e.g., a firewall or an IDS). This is accomplished via established communication methods of the particular "countermeasures":

> The countermeasure communications engine performs the key functionality for clientless NAC by driving firewalls, VPNs and SmartSwitches to be automatically reconfigured through remote control using their published application programming interfaces (APIs) through secure means such as OPSEC or authenticated SSH and command line interfaces.
> (Ex. 1006 at Figure 11).

106. Miliefsky also describes such integrations this way:

> FIGS. 10-17 illustrate security and vulnerability management on micro appliances, which comprises… clientless network admission control (CLIENTLESS NAC) integration with all major firewalls and smartswitch rules and access tables to quarantine problems (CVEs) at the network ports, whether physical or based on the Internet standard (TCP/IP) for ports, or similar protocol based software ports, where these problems reside.
> (Ex. 1006 at [0077]).

107. Such communication between the relevant "Engine" and the "countermeasure" may be facilitated by a "client plug-in." For example:

> In the event the INFOSEC countermeasure is a firewall or traffic filtering router, dynamic alerting of the IT manager or an alternative alert recipient and dynamic changes to the firewall rule table will take place through the Countermeasure Communications Client plug-in which has been written for that MAKE, MODEL and VERSION firewall or traffic filtering router. This may temporarily disable, turn off, or block network access either granularly through Port related CVE data or non-granularly by blocking all traffic of the said network-based asset containing the CVE(s) which need remediation.
> (Ex. 1006 at [0062]).

C. OVERVIEW OF US PATENT 8,104,089 TO GUO

108. U.S. Patent 8,104,089 to Fanglu Guo et al. ("Guo") is entitled "Tracking memory mapping to prevent packers from evading the scanning of dynamically created code." Guo explains that many malware authors create evasion systems for preventing their malware from being detected. For example, "A packer is a tool that is used to take an existing piece of malware and hide or 'pack' it to make it no longer detectable to most signature-based detection systems. A packer can change the byte-level appearance of a binary program… to hide the malware program by changing its appearance." (Ex. 1011 at 1:31-39). Furthermore, "more than 1000 different packers are currently known." (*Id.* at 1:49-50). Malware writers can pack malware with many different packers, perhaps even in combination, generating many variants of the same malware. (*See id.* at 3:44-49). Guo explains that it is directed toward detecting this kind of evasion: "Embodiments of the invention anticipate some of these evasion techniques and apply countermeasures to render those evasion techniques ineffective." (*Id.* at 2:36-39).

109. Part of the problem with the existing anti-packing systems is that the anti-virus companies must create a specific anti-packer for each packer. "Traditionally, AV companies have attempted to manage the packer problem by manually reverse-engineering popular packers and creating unpackers for them." (*Id.* at 1:43-45). This is simply not scalable: "Unfortunately, there are a very large number of different packers; more than 1000 different packers are currently known. Thus, reverse-engineering each of these different packers is an impossible proposition because it is a slow and expensive process." (*Id.* at 1:48-52). Using this large number of packers, sometimes in combination, "malware writers can easily generate a large number of variants from the same malware." (*Id.* at 3:50-51).

110. Instead of manually applying reverse-engineering via a per-packer solution, Guo provides a general, automated approach that reverse-engineers the software while it is running to

extract the original executable instructions. Guo further explains the advantages by stating that, "the packing manager 120 is generic because the way it tracks the code pages of the original programs ***does not require any packer-specific knowledge***." (*Id.* at 5:18-20, emphasis added). Moreover, "it is efficient because page status tracking can be done transparently by virtual memory hardware, and thus does not incur any software overhead." (*Id.* at 5:22-24).

111. Guo's techniques also detect multiple packing, wherein the data is packed multiple times, potentially by different packers:

> The packing manager system 120 can deal with multiple-layer packed binaries equally effectively without any modification. Since each layer of packer exhibits the same behavior, executing dynamically generated code, each layer will be unpacked in the reverse order of their packing sequence. If the latest unpacking round is not the last round, its output is an intermediate packed binary that needs to be unpacked. In this case, the program resumes and the packing manager 120 continues to track pages that are dynamically generated and executed.
> (*Id.* at 5:32-41).

112. Guo's approach is to process the application in real-time. Guo explains that "[t]o detect malicious code that is dynamically created (i.e., ***created at runtime***), e.g., by a packer, antivirus software may require ***dynamically created code to be scanned before it can be executed by a host computer system***." (*Id.* at 2:22-25, emphasis added). Guo does this by monitoring memory pages:

> Regardless of the packer used, the unpacker routine in a packed binary at some point creates the original program being packed at run time and transfers control to it. So, during the execution of a packed binary, the code pages associated with the original program must be dynamically generated and later on executed. By catching control transfers to dynamically written pages, the system described here can catch the moment when the original program is restored.
> (*Id.* at 4:26-34).

113. Guo explains one technique for tracking "transfers to dynamically written pages." (*Id.* at 4:35-36). This approach uses permissions on memory pages to catch the transfer. Guo

explains that a page can be marked as "writeable but non-executable." When the malware writes to the page and then tries to execute, it generates an event that the malware scanner can intercept. The data in the page is scanned. If it is malicious, the process is terminated. Otherwise, the execution can continue:

> More specifically, before an executable binary is executed in this embodiment, the packing manager system 120 marks its memory pages as non-writable but executable… When the unpacker starts restoring the original program, it will write to non-writable pages (marked NW by the packing manager) and will cause a write access violation exception for the first write to each such page… The virtual memory hardware automatically detects the write exception and the operating system delivers this exception to the user process. The packing manager 120 intercepts and handles each such exception by marking the associated page as writable but non-executable and resuming the execution so that the unpacker can continue restoring the original program…
>
> After the unpacker fully restores the original program, it transfers control to the entry-point (the first instruction) of the original program. The virtual memory hardware detects an execution exception because that page is writable but not executable. The operating system again delivers this exception to the user process. The packing manager 120 catches this exception and calls AV scanner to scan the address space snapshot at this point. For example, the system can dump the process's address space image for an AV scanner to scan. If the AV scanner detects malware, the packing manager 120 terminates the process.
> (*Id.* at 4:44-5:9).

114. As previously noted, this approach handles any number of packers layered together. (*See id.* at 5:32-34).

115. In summary, Guo teaches methods for using reverse-engineering techniques to analyze running code in order to detect obfuscation techniques (e.g., packing and multiple packing) and determine the malware status of such code.

IX. GROUND 1: CLAIMS 1-6 ARE RENDERED OBVIOUS BY MUKHERJEE IN VIEW OF MILIEFSKY

116. Claims 1-6 of the ’616 patent are rendered obvious to a POSITA by Mukherjee in view of Miliefsky, as explained in the sections below. I also explain that it would have been obvious for a POSITA to combine Mukherjee and Miliefsky.

A. Motivation to Combine Mukherjee and Miliefsky

117. A POSITA would have been motivated to combine Mukherjee and Miliefsky. 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. 30).

118. Miliefsky likewise a computer security system that can be used "… in defending against common vulnerabilities and exploits.” (Ex. 1006 at [0004]). Like Mukherjee, Miliefsky is also focused on vulnerabilities. While Mukherjee focuses on using vulnerabilities to detect threats, Miliefsky uses this information to mitigate and proactively block intruders.

> The system and methods described herein include, among other things, security systems that provide ***proactive automated defense against hackers by automatically finding, reporting, communicating with countermeasures about and removing the common vulnerabilities and exposures (CVEs) that they exploit***. Accordingly, the systems described herein provide for proactive security by determining the components that exist on a network system and generating a list of network assets.
> (Ex. 1006 at [0039]).

119. A POSITA would have recognized that the two systems are naturally compatible with each other. Mukherjee explains that the DIDS system is meant to be used in concert with other systems specifically to obtain real-time defenses.

> ***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).

120. Moreover, as mentioned above, Mukherjee already recognizes that IDS systems use vulnerability rules in their operation.

121. Miliefsky also incorporates IDS systems, such as Mukherjee's DIDS, into its own operation. Miliefsky explains that the security appliance includes INFOSEC (Information Security) components and plugins of various types including IDS.

> In the event the INFOSEC countermeasure is an IDS, dynamic alerting of the IT manager or an alternative alert recipient and ***sharing the related CVE tests data with the IDS to help the IDS reduce false positives in the IDS alerting module as well as reduce the traffic load*** related to intrusion detections which attack a particular IP address that is not susceptible to that particular attack methodology based upon the related CVE tests data and will take place through the Countermeasure Communications Client plug-in which has been written specifically for that MAKE, MODEL and VERSION IDS.
> (Ex. 1006 at [0064]).

122. As I noted in Miliefsky's teachings above, Miliefsky teaches why a POSITA would have been motivated to combine Miliefsky's data into an IDS's operation, such as the DIDS system taught by Mukherjee. First, Miliefsky's teachings would "help the IDS reduce false positives." (*Id.*). Second, it would "reduce the traffic load." (*Id.*). A POSITA also would have known that any of Miliefsky's proactive fixes, which prevent an attacker from getting into the system at all, would also reduce the load on the IDS.

123. Thus, a POSITA would have been motivated to combine Miliefsky with Mukherjee in order to fulfill Mukherjee's goals of real-time response to attackers while, at the same time,

incorporating Miliefsky's third-party threat data (e.g., CVEs) in order to reduce false positives and the traffic load on the IDS system. Moreover, given that Mukherjee discloses that integration is expected and desired, and Miliefsky discloses that IDS incorporation is expected and desired, it would be obvious to try and with a high expectation of success.

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

124. Each of the limitations of Claim 1 of the '616 patent is rendered obvious to a POSITA by Mukherjee in view of Miliefsky as described below.

1. Limitation 1[a]

125. Limitation 1[a] of the '616 patent reads:

> *A method of providing an attestation service for providing runtime operational integrity of a system using a computing platform comprising a network trust agent, an endpoint trust agent, and a trust orchestration server, the method comprising:*

126. 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 limiting, Mukherjee in view of Miliefsky discloses a method of providing an attestation service for providing runtime operational integrity of a system using a computing platform comprising a network trust agent, an endpoint trust agent, and a trust orchestration server.

127. Mukherjee discloses "an attestation service for providing runtime operational integrity of a system". Mukherjee explains a system that can watch activities on hosts and receive events generated by the host.

> 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 expert system is a rule-based system with simple learning capabilities. 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).
>
> 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.
> (*Id.* at p. 11-12, emphasis added).

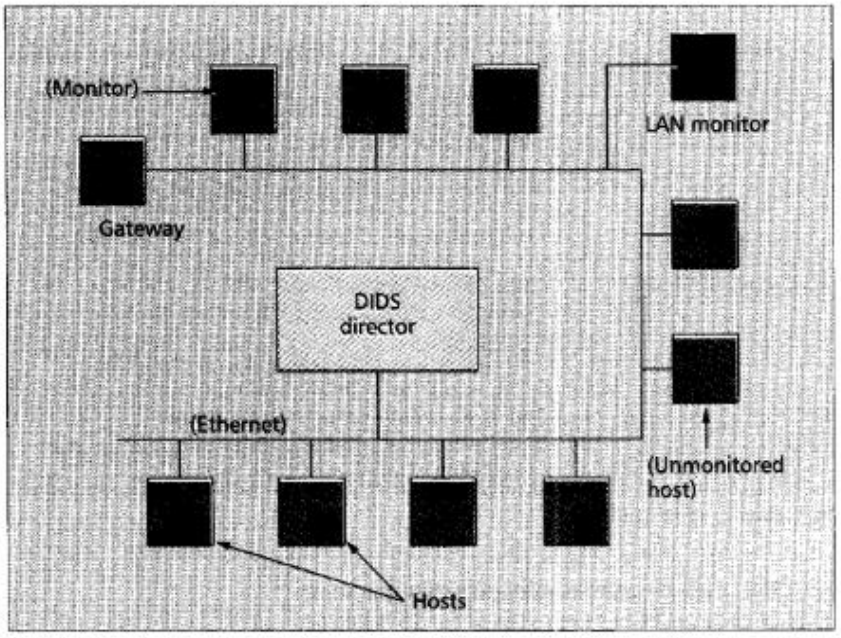
128. Mukherjee explains that the System Security Officer (SSO) can watch activities on each host and request specific types of information from the monitors. The SSO is described as the user to the "user interface" of the DIDS system, wherein the "DIDS director consists of three major components that are all located on the same dedicated workstation – a communications manager, an expert system, and a user interface." A POSITA would have understood dynamically reporting activities that pertain to the integrity of the system to be "providing a runtime operational integrity of a system."

129. The monitors described by Mukherjee are stated to report interesting events to a centrally located DIDS director.

> ***In DIDS, the host and LAN monitors report any "interesting" events, which may possibly lead to intrusive activity, to a centrally-located DIDS director. The director employs an expert system to detect possible attacks.*** An initial prototype of DIDS has successfully demonstrated its ability to track users as they move around the network (possibly attempting to hide their true identities) and to identify doorknob rattling attacks [32,34]. 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.
> (Ex. 1005 at p. 11).

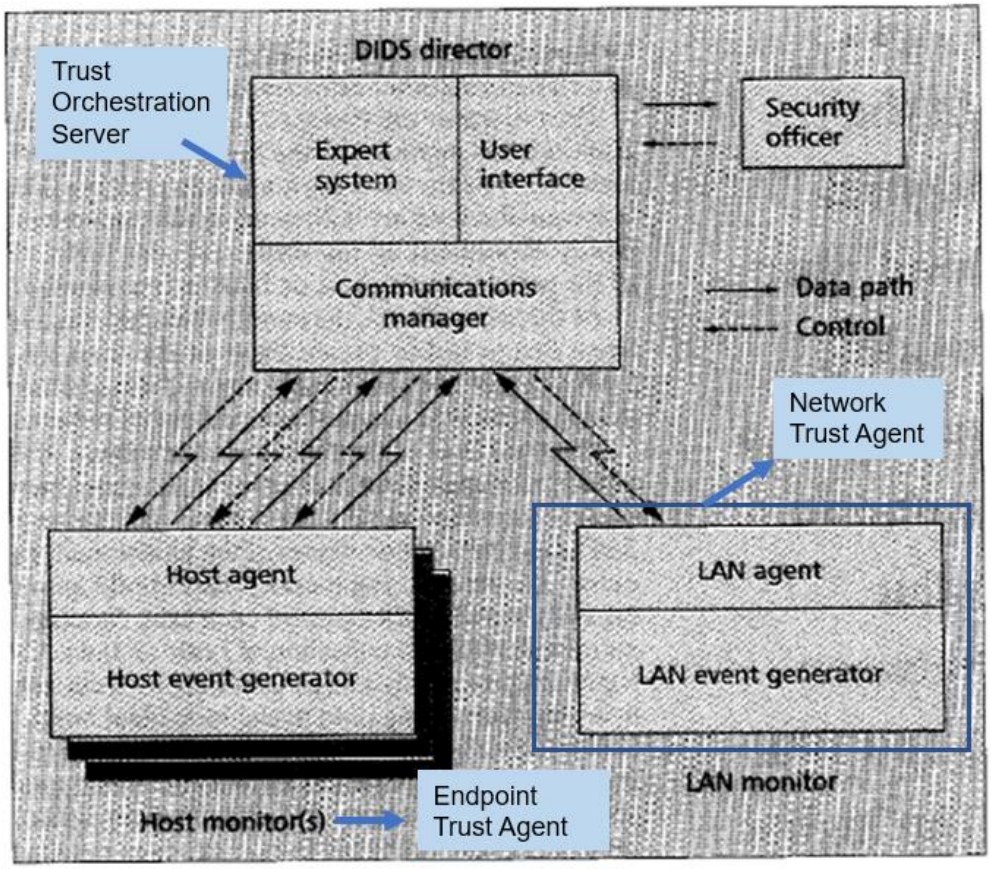
130. A POSITA would have understood reporting events which lead to intrusive activity and employing a director to detect possible attacks to be an assessment at runtime.

131. Mukherjee discloses a "computing platform comprising a network trust agent, an endpoint trust agent, and a trust orchestration server." For example, Mukherjee discloses a LAN monitor, a Host monitor, and a DIDS director. These components are visualized in Figures 1 and 2 (annotated to indicate mappings to '616 disclosures) reproduced hereinbelow:



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

(Ex. 1005 at Figure 1).



(Ex. 1005 at Figure 2, annotated).

132. A POSITA would have recognized a LAN Monitor as a "network trust agent," a Host Monitor as an "endpoint trust agent," and a DIDS director as a "trust orchestration server." I discuss each of these three components in detail below.

Network Trust Agent

133. With respect to a "network trust agent," the '616 patent does not provide a definition. However, given the terms used, a POSITA would have understood a "network trust agent" to mean a component that determines, measures, reports on, or ensures the trustworthiness of a network or the actors or devices that interact on such network. The LAN Monitor disclosed by Mukherjee determines network trustworthiness by observing network events useful in identifying intrusion behavior.

> 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.
> (*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).
>
> …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.
> (*Id.* at p. 13, emphasis added).

134. The '616 specification, although not providing a definition of "network trust agent," describes embodiment in which a network trust agent can communicate with other network devices to enforce network access. Specifically, the network trust agent can send "directives" to a controller that diverts or blocks network traffic.

> In step 2107, ***the network trust agent 1209 may generate and send to the computer network protocol (e.g., OpenFlow™) security framework 1211 directives 1210 to formulate commands*** 1213 for the computer network protocol (e.g. OpenFlow™) controller 1214.
>
> In step 2108, the computer network protocol (e.g., OpenFlow™) controller 1214 sends rules 1215 to the computer network protocol (e.g., OpenFlow™)

> enabled network element (switch) 1216 ***to divert or block traffic flows to/from the forewarned network device 560***.
> (Ex. 1001 at 33:7-15, emphasis added).

135. The network trust agent can also reset the device to "normal flows."

> In step 2112, the network trust agent 1209 may generate and send to the computer network protocol (e.g., OpenFlow™) security framework 1211 directives 1210 to formulate commands 1213 for the computer network protocol (e.g., OpenFlow™) controller 1214 to restore normal flows to/from the forewarned network device 560.
> (Ex. 1001 at 33:34-39).

136. To the extent that a "network trust agent" needs to be able to directly or indirectly control network flows, Miliefsky discloses a "network trust agent." Miliefsky discloses a countermeasure system that includes a "communications engine." This system is designed to communicate with other components with functions to terminate communications or otherwise trigger additional network security checks.

> A client-server modular based software system for secure, authenticated and non-reputable communications between the Proactive Network Security system and any traditional or typical Countermeasures System to increase the probability that a hacker will not be able to break into the existing network infrastructure through automated vulnerability assessment, reporting, and remediation. ***The Countermeasures Communications System enables secure communications between the Anti-Hacker Proactive Network Security System and other computer-based network equipment*** which may be newly designed or traditional INFOSEC countermeasure solutions such as stateful packet inspecting firewall, virtual private networking through IP Security (IPSec), Secure Sockets Layer (SSL) to Intrusion Detection, Intrusion Prevention, Honeypot, Anti-Virus, to Anti-Spam and other countermeasures-based INFOSEC functionality not originally built into the Anti-Hacker system design. The Countermeasure Communications System is structured to enable secure communications between the Anti-Hacker Proactive Network Security System and other computer-based network equipment which may be newly designed or traditional INFOSEC countermeasure solutions. ***Functions are available to initiate and terminate communications, allow the INFOSEC countermeasure client to initiate requests for scheduling or immediate vulnerability assessments through CVE tests, request reports in pre-defined file formats or a data feed of the results, request remediation on one, more or all of the IP Addresses which were tested or scheduled to be***

> ***tested and to request dynamic updates to client INFOSEC countermeasure system***.
> (Ex. 1006 at [0055], emphasis added).

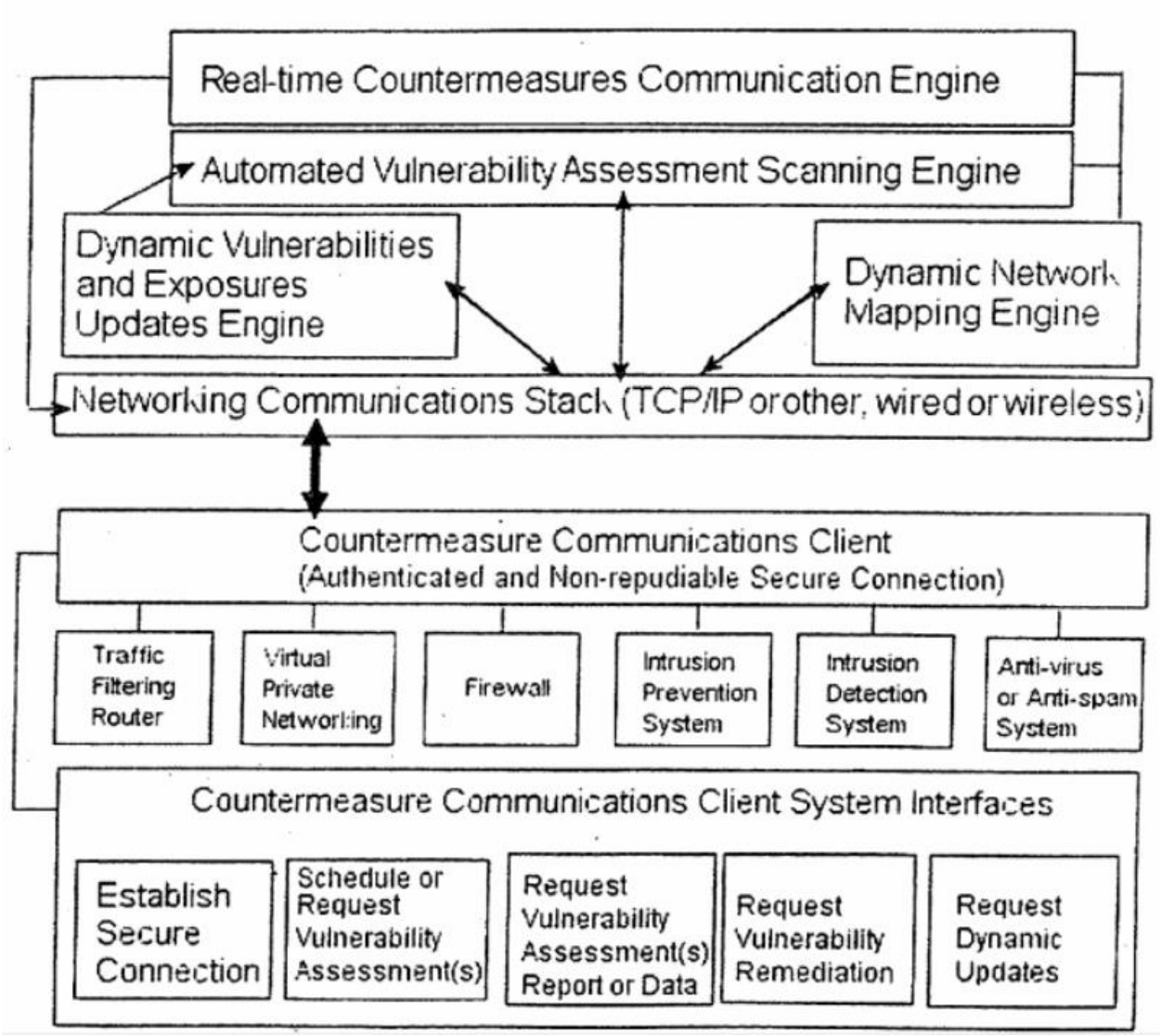
137. The countermeasure communications engine controls other network systems such as firewalls, VPNs, and SmartSwitches. See, e.g., Ex. 1006 at Figure 11, a part of which is reproduced herein below:

> f. The countermeasure communications engine performs the key functionality for clientless NAC by driving firewalls, VPNs and SmartSwitches to be automatically reconfigured through remote control using their published application programming interfaces (APIs) through secure means such as OPSEC or authenticated SSH and command line interfaces.

138. Network components interface with the communications engine through, for example, "Countermeasure Communications Client plug-ins." These plug-ins can be used to change the rules in a firewall.

> In the event the INFOSEC countermeasure is a firewall or traffic filtering router, dynamic alerting of the IT manager or an alternative alert recipient and ***dynamic changes to the firewall rule table will take place through the Countermeasure Communications Client plug-in*** which has been written for that MAKE, MODEL and VERSION firewall or traffic filtering router. ***This may temporarily disable, turn off, or block network access either granularly through Port related CVE data or non-granularly by blocking all traffic of the said network-based asset containing the CVE(s) which need remediation.***
> (Ex. 1006 at [0062], emphasis added).

139. See also, for example, Ex. 1006 Figure 7, reproduced herein below.



(Ex. 1006 at Figure 7).

140. Miliefsky explains that one purpose of the system is to "quarantine" untrusted network assets.

> ***The Countermeasure Communications engine shares dynamically detected information about current and new network assets for the dynamic reconfiguration of firewalls, virtual private networks (VPNs) and SmartSwitches to quarantine CVEs (problems) detected in any and all trusted network assets at the port level, blocking problems at ports, not people and productivity. In the event a network asset is untrusted, such as a rogue laptop or wireless router, it is quarantined at all possible points of entry and exit including but not limited to the firewall, VPN and SmartSwitch.*** This system will also send an alert through E-mail and SMS

> paging to an IT Manager or designated end user to let them know that the system detected a rogue or high risk asset and took action, automatically.
> (Ex. 1006 at [0086], emphasis added).

Endpoint Trust Agent

141. The '616 patent describes the "endpoint trust agent" as an application that component or application that can assess operational integrity based on rulesets. The agent can observe the operation of other applications on the system in terms of execution context and network activities. It also can use resource utilization monitors.

> As shown in FIG. 5, the exemplary application operational integrity system 500 includes an endpoint trust agent 510 on a device 560 comprising a process monitor 514 ***configured to observe local execution context of applications and services***. The endpoint trust agent 510 further comprises 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***. In the example embodiment provided in FIG. 5, the trust agent 510 also comprises an application integrity module 516 and a resource utilization module 515 ***configured to assess operational integrity based on rulesets***. The trust agent 510 can also comprise native machine instrumentation 511 for a computing device being monitored by the application operational integrity system 500.
> (Ex. 1001 at 13:4-18).

142. Mukherjee discloses an "endpoint trust monitor" in the form of a host monitor. The host monitor includes a "host event generator (HEG)" that collects audit data from the 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).

143. A POSITA would have understood that audit records from the OS provide execution context for running applications and services.

144. Mukherjee also discloses that the host monitor tracks network access. 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). A POSITA would have understood that the host monitor is configured to monitor network activities.

145. Additionally, the host monitor, as disclosed by Mukherjee, also tracks behavioral measures of activity on the system. This includes "[a]bout two dozen 'features' (behavioral measures) of the user's session… [such as] time of work, number of files created, number of pages printed, etc." (Ex. 1005 at p. 5). This analysis is part of the HAYSTACK algorithm. As Mukherjee explains, "[t]he HEG also tracks user sessions and report [sic] 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, emphasis added).

146. Thus, a POSITA would have understood the host monitor to be an "endpoint trust agent."

Trust Orchestration Server

147. The '616 specification only mentions "trust orchestration server" once outside of the claims but appears to equate it with "trust orchestrator."

> FIG. 21 is a flowchart illustrating steps by which an attestation service for runtime operational integrity of systems executing on a computing platform using a network trust agent, endpoint trust agent and ***trust orchestration server*** can be provided. The steps of the method 2100 shown in FIG. 21 do not necessarily have to occur in the order described.
>
> Referring to FIG. 21, the method 2100 can provide network flow level remediation for an infected device 560 at a network element 1216 ***using a trust orchestrator 1205,*** an endpoint trust agent 1201, a network trust agent 1209 and collaboration services 1207.
> (Ex. 1001 at 32:45-55, emphasis added).

148. The ’616 patent further explains that the system, in one embodiment, includes “[w]ith continued reference to the exemplary embodiment provided in FIG. 6, the trust orchestrator 430 comprises a system event correlator 410 configured to use a risk correlation matrix 411 to generate an integrity profile 420 for the monitored system (i.e., the target system).” (Ex. 1001 at 15:32-40).

149. Mukherjee discloses a system event correlator in the form of the DIDS director.

> 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***…
> (Ex. 1005 at p. 12, emphasis added).

150. Moreover, the DIDS director comprises an expert system that correlates system events. The different levels of the Intrusion Detection Model (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). In particular, Mukherjee discloses correlating events in the “fourth layer” of the IDM.

> 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 emphasis added).

151. Thus, a POSITA would have recognized the DIDS director as a Trust Orchestration Sever.

152. A POSITA would have been motivated to combine Mukherjee and Miliefsky. I have already explained the general motivation for this combination in Section IX.A. I incorporate this analysis into this section by reference.

153. More specifically, for this limitation, a POSITA would have been motivated to incorporate Miliefsky's countermeasure communications engine into Mukherjee's IDS in order to prevent damage by an attacker to the system. As I have already explained, Mukherjee explains this is a goal of the broader effort, even though the DIDS system does not have the functionality natively.

> ***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).

154. Thus, a POSITA would have been motivated to extend Mukherjee's capabilities to actively respond to attacks. A POSITA would have been motivated to use Miliefsky's disclosures because Miliefsky is designed, as discussed above, to plug into many systems including IDS systems such as Mukherjee's DIDS systems. Miliefsky, as also discussed above, includes a plug-in based architecture to extend functionality to many systems. As plug-in architectures were well known and widely deployed, it would have been obvious for a POSITA to try combining

Miliefsky's disclosures with Mukherjee's, and they would expect minimal experimentation and a high chance of success.

155. Mukherjee in view of Miliefsky discloses the rest of the steps of the method as discussed in the paragraphs below.

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

2. Limitation 1[b]

157. Limitation 1[b] of the '616 patent reads:

> *sending, by the endpoint trust agent on a monitored device, a dynamic context including endpoint events and actions of the monitored device and applications executing on the monitored device at runtime;*

158. Mukherjee discloses limitation 1[b].

159. Mukherjee discloses "sending, by the endpoint trust agent on a monitored device, a dynamic context including endpoint events and actions of the monitored device and applications executing on the monitored device at runtime." Mukherjee's host monitor sends notable events, such as "failed events", and actions, such as *rlogin* and *rsh* network access, to the DIDS director.

> 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 p. 12, emphasis added).

160. A POSITA would have understood that the collection of endpoint actions and events are a "dynamic context." The '616 patent specification expresses that a dynamic context may "include the local runtime context of an application 1203…" (Ex. 1001 at 25:27-35). A POSITA would have understood that the notable events disclosed by Mukherjee represent a runtime context of applications. For example, the user authenticated is part of the context of a launched application, as is the security state of the system and whether or not the user is remote or local.

161. As I explained previously in limitation 1[a], the endpoint trust agent in the '616 patent would have been understood by a POSITA to be the host monitor in Mukherjee's system.

162. Mukherjee also discloses that this process occurs at runtime. As explained with respect to limitation 1[a], Mukherjee discusses identifying security issues 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, emphasis added).

163. A POSITA would have known that distinguishing between binaries and running processes constitutes real-time operations.

164. Furthermore, as I have also explained with respect to limitation 1[a], a POSITA would have recognized that the DIDS director is a trust orchestration server. Mukherjee explains that the host monitor sends the events and actions to the DIDS director.

> 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).

165. Thus, I conclude that Mukherjee discloses limitation 1[b].

3. Limitation 1[c]

166. Limitation 1[c] of the ’616 patent reads:

receiving, at the trust orchestration server, the dynamic context including the endpoint events of the monitored device and the applications executing on the monitored device at runtime;

167. Mukherjee discloses limitation 1[c].

168. Mukherjee discloses “receiving, at the trust orchestration server, the dynamic context including the endpoint events of the monitored device and the applications executing on the monitored device at runtime.” I have already explained, with respect to limitation 1[b], that the dynamic context is transmitted to the DIDS director, which is the trust orchestration server. Mukherjee discloses that the DIDS director receives the events and actions that comprise the dynamic context.

> 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).

169. Thus, I conclude that Mukherjee discloses limitation 1[c].

4. Limitation 1[d]

170. Limitation 1[d] of the ’616 patent reads:

analyzing, by the trust orchestration server, the received endpoint events;

171. Mukherjee discloses limitation 1[d].

172. Mukherjee discloses "analyzing, by the trust orchestration server, the received endpoint events." As I explained in limitations 1[a] – 1[c], A POSITA would have understood Mukherjee's DIDS director to be a trust orchestration server. I also explained with respect to limitation 1[c] that the dynamic context of actions and events is received by the DIDS director.

173. Mukherjee further explains that the endpoint events that are sent by the host monitor are analyzed.

> 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*.**
> (Ex. 1005 at p. 12, emphasis added).
>
> 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.***
> (*Id.* at p. 11, emphasis added).
>
> ***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.* at p. 12, emphasis added).

174. A POSITA would have understood that making inferences about the security of a host or system based on received endpoints events is a form of analysis. Furthermore, an evaluation of "evidence of unauthorized or suspicious activity" is additionally understood to be an analysis.

175. Thus, I conclude that Mukherjee discloses limitation 1[d].

5. Limitation 1[e]

176. Limitation 1[e] of the '616 patent reads:

receiving, by the trust orchestration server, third party network endpoint assessments;

177. Mukherjee, or Mukherjee in view of Miliefsky, discloses limitation 1[e].

178. Mukherjee discloses "receiving, by the trust orchestration server, third party network endpoint assessments." Mukherjee explains that the DIDS system receives information from a third party network endpoint assessment, namely Network Security Monitoring (NSM) Services. NSM services provide network assessment to clients.

> 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).

179. A "network endpoint assessment" was well understood in the art. It was so common that the acronym "NEA" was in use by 2008 and there were known reference models. An RFC (Request For Comment) from 2008, for example, states, "[t]his document defines the problem statement, scope, and protocol requirements between the components of the NEA (Network Endpoint Assessment) reference model." (Ex. 1020 at p. 1). In this reference model, "NEA typically involves the use of special client software running on the requesting endpoint that observes and reports on the configuration of the system to the network infrastructure." (*Id* at p. 3). The NEA client communicates with an NEA Server. (*Id.* at pp. 15-17).

180. A POSITA would have understood a report sourced from Mukherjee's NSM services to be a "third party network endpoint assessment." The NSM servers are remote from the DIDS system and are not directly involved with the transactions between the DIDS director and

the host monitor. Therefore, the NSM services are a third party. Finally, NSM services provide network assessments to clients.

181. Moreover, Mukherjee explicitly endorses transmitting audit records from third-party packages.

> 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.
> (Ex. 1005 at p. 13, emphasis added).

182. In my opinion, the plain and ordinary meaning of "third party network endpoint assessments" refers to network endpoint assessments created by others. Mukherjee discloses a "third-party auditing package" which is a package from another vendor that provides information a POSITA would have recognized as teaching "endpoint assessments."

183. Thus, Mukherjee discloses "receiving, by the trust orchestration server, third party network endpoint assessments." In addition, Mukherjee in view of Miliefsky discloses this limitation.

184. Miliefsky discloses a "third party network endpoint assessment" that is a service provided by a different vendor. Specifically, Miliefsky discloses a network security apparatus that incorporates data from many components as well as "third party sources."

185. For example, Miliefsky discloses that the security appliance collects a wide range of data including vulnerability assessments from INFOSEC plugins. These assessments use CVEs, which are third-party vulnerability information, to make the assessments:

> An Information Security (INFOSEC) software engine acts as a gateway between users, personal computers, servers, services and the computer network (internet, intranet, extranet, wide area network, wireless network or local area network). The information Security (INFOSEC) Engine controls the computer-based network scanning, standards-based

> vulnerability assessment through common vulnerabilities and exposures (CVEs) testing, reporting and remediation as well as interfacing with the INFOSEC ENGINE PLUGINs. The INFOSEC Engine is structured in a modular fashion with a main controller that takes input for control from the Secure GUI modules. Functions include reading and acting upon the configuration and scheduling data as stored by the Secure GUI modules. ***The INFOSEC Engine contains a unique module for each vulnerability assessment CVE test as well as communication modules to enable non-intrusive testing for each unique IP Address accessible from the computer-based network. The INFOSEC Engine contains read, write and export functionality for vulnerabilities found*** and reported in various formats including but not limited to structured query language (SQL) databases and tables, portable document format (PDF), extensible markup language (XML), hypertext markup language (HTML), comma separated values (CSV) and Excel file format (XSL). The INFOSEC Engine, at initialization, is able to determine which CVE tests are available as well as which INFOSEC Engine Plugins are available and then to relay this information to the Secure GUI for administration, control and management. (Ex. 1006 at [0049], emphasis added).

186. Miliefsky further explains that assessment data is collected using a standard language called OVAL. OVAL permits a standardized format for communication within the security community.

> OVAL is the Open Vulnerability Assessment Language. It is funded by the U.S. Department of Homeland Security (DHS) and in summary the XML, machine readable format for the Common Vulnerabilities and Exposures (CVE®) standard. OVAL is an international, information security community baseline standard for how to check for the presence of vulnerabilities and configuration issues on computer systems. OVAL standardizes the three main steps of the process with an OVAL System Characteristics Schema for collecting configuration data from systems for testing; OVAL Definitions to test for the presence of specific vulnerabilities, configuration issues, and/or patches; and an OVAL Results Schema for reporting the results from the evaluated systems. The tests are standardized, machine-readable XML Vulnerability Definitions, Compliance Definitions, and Patch Definitions. OVAL's schemas and definitions are all free to download, use, reference, and implement. An "OVAL-compatible" tool, service, Web site, database, or advisory/alert uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state. An "OVAL-ID compatible" tool, Web site, database, archive, or security advisory includes OVAL-ids as part of the information it

> conveys about a security issue, and provides for searching by OVAL ID with potential linkage back to the source definition of the OVAL-ID.
> (Ex. 1006 at [0103]-[0104]).

187. However, Miliefsky then identifies that the database that holds results for this system can include external, third-party assessments in addition to its own.

> Database Correlation and Warehousing Engine: ***This engine gathers data from processes and results throughout the product as well as from internal/external resources, including but not limited to the Update Servers, Countermeasure appliances, Data feeds, other devices of the same nature as this patent describes and any related third party sources***. The engine uses data warehouse methodologies to store this data. The engine also provides a means of querying the database and warehouse information either through automated methods or through on-demand user interfaces.
> (Ex. 1006 at [0093], emphasis added).

188. Miliefsky also notes that compatible vulnerability assessment systems that use OVAL can provide data to Miliefsky's system. Such systems can include a "Web site," which a POSITA would have understood to include web sites hosted by third parties:

> ***An "OVAL-compatible" tool, service, Web site, database, or advisory/alert*** uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state. ***An "OVAL-ID compatible" tool, Web site, database, archive, or security advisory*** includes OVAL-ids as part of the information it conveys about a security issue, and provides for searching by OVAL ID with potential linkage back to the source definition of the OVAL-ID.
> (Ex. 1006 at [0104], emphasis added).

189. In fact, Miliefsky's architecture is designed around a system that incorporates components created by other vendors. As Miliefsky explains:

> A client-server modular based software system for secure, authenticated and non-reputable communications between the Proactive Network Security system and ***any traditional or typical Countermeasures System*** to increase the probability that a hacker will not be able to break into the existing network infrastructure through automated vulnerability assessment, reporting, and remediation. The Countermeasures Communications System enables secure communications between the Anti-Hacker Proactive

> Network Security System and ***other computer-based network equipment which may be newly designed or traditional INFOSEC countermeasure solutions***…
> (Ex. 1006 at [0055], emphasis added).

190. Thus, Miliefsky discloses "receiving… third party network endpoint assessments." Moreover, Mukherjee in view of Miliefsky discloses "receiving, by the trust orchestration server, third party network endpoint assessments."

191. I have already explained in Section IX.A. the general motivations a POSITA would have had to combine Mukherjee and Miliefsky, and I incorporate this analysis into this section by reference.

192. Specifically with reference to this limitation, a POSITA would have been motivated to combine Miliefsky's network assessments with Mukherjee's IDS system. As explained in Section IX.A., Miliefsky already describes combining assessments with IDS's and notes the advantages.

> In the event the INFOSEC countermeasure is an IDS, dynamic alerting of the IT manager or an alternative alert recipient and ***sharing the related CVE tests data with the IDS to help the IDS reduce false positives in the IDS alerting module as well as reduce the traffic load*** related to intrusion detections which attack a particular IP address that is not susceptible to that particular attack methodology based upon the related CVE tests data and will take place through the Countermeasure Communications Client plug-in which has been written specifically for that MAKE, MODEL and VERSION IDS.
> (Ex. 1006 at [0064]).

193. Thus, Miliefsky teaches why a POSITA would have been motivated to combine Miliefsky's data into an IDS's operation. First, it would reduce false positives. Second, it would reduce the traffic load. A POSITA also would have known that any of Miliefsky's proactive fixes, which prevent an attacker from getting into the system at all, would also reduce the load on the IDS.

194. Thus, I conclude that Mukherjee, or Mukherjee in view of Miliefsky, discloses limitation 1[e].

6. Limitation 1[f]

195. Limitation 1[f] of the ’616 patent reads:

> *generating, by the trust orchestration server, temporal events based at least in part on analyzing the third party network endpoint assessments;*

196. Mukherjee in view of Miliefsky discloses limitation 1[f].

197. The specification for the ’616 patent explains that temporal events can be “expressed as assertions about operational parameters (e.g., vulnerabilities, compliance, patch level, etc.) based on enterprise policies established for a baseline configuration.” (Ex. 1001 at 11:52-55). It states further,

> With continued reference to FIG. 4, the temporal events 409 ***can be an annotation (e.g., an XML representation) of the configuration state of the system*** (including device, application and package) and include trust and severity scores as generated by third party security vendors based on integrity measurement and verification, in accordance with standards established by organizations… ***temporal events can be assertions about operational parameters expressed in the Open Vulnerability and Assessment Language (OVAL) to convey system details***…
> (Ex. 1001 at 11:59-12:16, emphasis added).

198. Miliefsky discloses “generating… temporal events.” For example, Miliefsky discloses generating lists of equipment assets and their configuration.

> ***As part of the security method, in a first step the network is scanned and/or probed for any and all attached equipment and related assets, herein referred to as “network-based” assets***. The method will dynamically detect and map changes to LAN and WAN connected equipment including searching for equipment which may be deemed as rogue and creating a network-based assets list, wherein the list contains information as to the location of the network-based assets.
>
> ***The list may contain information as to the Internet Protocol (IP) address of said network-based assets, and the list may contain information as to***

> ***the open ports of said network-based assets and related application, session, transport, sockets and other internet protocol (IP) related information. The list may contain other information such as the Media Access Control (MAC) address of said network-based assets, whether the connection is Wired or Wireless of said network-based assets and other information about the structure of the network and its component devices***. (*Id.* at [0040]-[0041]).

199. Miliefsky also discloses that the lists are updated on regular time intervals.

> The information contained in the list may change automatically and at pre-scheduled intervals as network-based assets are moved or relocated. (*Id.* at [0042]).

200. As mentioned hereinabove, the specification of the ’616 patent specifically calls out OVAL as a format for storing temporal event data. Miliefsky teaches collecting assessment data using this language:

> OVAL is the Open Vulnerability Assessment Language. It is funded by the U.S. Department of Homeland Security (DHS) and in summary the XML, machine readable format for the Common Vulnerabilities and Exposures (CVE®) standard. OVAL is an international, information security community baseline standard for how to check for the presence of vulnerabilities and configuration issues on computer systems. ***OVAL standardizes the three main steps of the process with an OVAL System Characteristics Schema for collecting configuration data from systems for testing; OVAL Definitions to test for the presence of specific vulnerabilities, configuration issues, and/or patches; and an OVAL Results Schema for reporting the results from the evaluated systems.*** The tests are standardized, machine-readable XML Vulnerability Definitions, Compliance Definitions, and Patch Definitions. OVAL's schemas and definitions are all free to download, use, reference, and implement. An “OVAL-compatible” tool, service, Web site, database, or advisory/alert uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state. (Ex. 1006 at [0103]-[0104]).

201. Miliefsky also discloses “generating… based at least in part on analyzing the third party network endpoint assessments.” First, the configuration data, Miliefsky explains, can be combined with vulnerability assessment data based on CVEs in order to protect against attacks.

> In a further step, the method audits one or more of the network-based assets for common vulnerabilities and exposures (CVEs) as defined by the U.S. federally funded CVE list managed by MITRE corporation or any similar list. The method will generate a CVE and related regulatory compliance audit reports and update the CVE and related regulatory compliance audit tests. ***In a further step the method can share MAC, IP, Port, CVE and related regulatory compliance other related audit data with various INFOSEC countermeasures designed to help protect network-based assets against attacks.***
> (*Id.* at [0043]).

202. And, as explained with reference to limitation 1[e], Miliefsky then identifies that the database that holds results for this system can include external, third-party assessments in addition to its own.

> Database Correlation and Warehousing Engine: ***This engine gathers data from processes and results throughout the product as well as from internal/external resources, including but not limited to the Update Servers, Countermeasure appliances, Data feeds, other devices of the same nature as this patent describes and any related third party sources***. The engine uses data warehouse methodologies to store this data. The engine also provides a means of querying the database and warehouse information either through automated methods or through on-demand user interfaces.
> (*Id.* at [0093], emphasis added).

203. The combination of assessments generated internally and assessments generated externally are used in determining if a specific host is safe to use the network.

> Clientless network admission control system: This engine provides a means to control the access of network devices onto networks. The engine does not require any software to be installed on any of the target devices. ***The engine uses a combination of the network discovery engine, vulnerability assessment engine, database correlation engine, wireless and mobile device detection engine to determine when a network device has permission to access the network***. This determination is also based upon information obtained from the regulatory compliance reviewing and reporting system and policies. This engine interacts with the countermeasure communications system to control the access of each network appliance. The engine is designed to work in a multi-branch solution and provide extensible authorization. ***It securely connects to firewalls, smart switches, and VPNs to reconfigure their rules and access***

> ***control lists around CVE related problems and ports***, not people and productivity.
> (Ex. 1006 at [0094]).

204. As I explained with respect to previous limitations, such as 1[a], a POSITA would have recognized the DIDS director as a trust orchestration server. It would have been obvious to a POSITA to modify the DIDS Intrusion Detection Model to incorporate Miliefsky's assessments into the operations of the DIDS director.

205. I have already identified the motivation for a POSITA to combine Mukherjee and Miliefsky in Section IX.A. I incorporate that analysis into this section by reference.

206. Specifically with reference to this limitation, a POSITA would have been motivated to combine Miliefsky's network assessments with Mukherjee's IDS system. As explained in Section IX.A., Miliefsky anticipates combining assessments with IDS's and notes the advantages.

> In the event the INFOSEC countermeasure is an IDS, dynamic alerting of the IT manager or an alternative alert recipient and ***sharing the related CVE tests data with the IDS to help the IDS reduce false positives in the IDS alerting module as well as reduce the traffic load*** related to intrusion detections which attack a particular IP address that is not susceptible to that particular attack methodology based upon the related CVE tests data and will take place through the Countermeasure Communications Client plug-in which has been written specifically for that MAKE, MODEL and VERSION IDS.
> (Ex. 1006 at [0064]).

207. Thus, Miliefsky teaches why a POSITA would have been motivated to combine Miliefsky's data into an IDS's operation. First, it would reduce false positives. Second, it would reduce the traffic load. A POSITA also would have known that any of Miliefsky's proactive fixes, which prevent an attacker from getting into the system at all, would also reduce the load on the IDS.

208. Thus, I conclude that Mukherjee in view of Miliefsky discloses limitation 1[f].

7. Limitation 1[g]

209. Limitation 1[g] of the ’616 patent reads:

correlating, by the trust orchestration server, the received endpoint events and the generated temporal events;

210. Mukherjee in view of Miliefsky discloses limitation 1[g].

211. Mukherjee in view of Miliefsky discloses “correlating, by the trust orchestration server, the received endpoint events and the generated temporal events.”

212. As I explained in previous limitations, including 1[a], a POSITA would have recognized the DIDS director as a trust orchestration server. I also explained with respect to limitation 1[f] that Mukherjee in view of Miliefsky discloses generating the temporal events required by that claim limitation. I incorporate my analysis of both of these limitations into this section by reference.

213. Mukherjee already discloses correlating events, such as system events and network events.

> 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***…
> (Ex. 1005 at p. 12, emphasis added).

214. Moreover, the DIDS director comprises an expert system that correlates system events. The different levels of the Intrusion Detection Model (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). In particular, Mukherjee discloses correlating events in the “fourth layer” of the IDM.

> 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, emphasis added).

215. Also, Mukherjee explains that the events generated may be produced by a third-party package, rather than the host or LAN monitors.

> …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, emphasis added).

216. It would have been obvious to a POSITA to include the temporal events I discussed with respect to limitation 1[f] into the correlation operations of Mukherjee. As I explained above, Mukherjee already discloses using third-party audit packages.

217. As I also explained with respect to limitation 1[f], a POSITA would have been motivated to combine Miliefsky's combination of asset data and assessment data with Mukherjee's DIDS director. It would also have been obvious to combine this data with the regular endpoint event data processed by the DIDS director.

218. I have already explained in Section IX.A. that a POSITA would have been motivated to combine Mukherjee and Miliefsky generally. I incorporate that analysis into this section by reference.

219. More specifically, with respect to this limitation, a POSITA would have been motivated to combine Miliefsky's baseline and configuration analysis, which are temporal events as required by this claim limitation, into Mukherjee's IDS analysis. As I have already explained, Mukherjee expects to use third-party audits. A POSITA would have been motivated to use Miliefsky to incorporate Miliefsky's advantages of reduced false positives and reduced IDS traffic. Moreover, because Miliefsky was designed to work with IDSs, a POSITA would have found it the combination required minimal experimentation with high confidence of success.

220. Thus, I conclude that Mukherjee in view of Miliefsky discloses limitation 1[g].

8. Limitation 1[h]

221. Limitation 1[h] of the '616 patent reads:

> *generating, by the trust orchestration server, an integrity profile for the system.*

222. Mukherjee and Miliefsky each disclose limitation 1[h]

223. Mukherjee discloses "generating, by the trust orchestration server, an integrity profile for the system." I have already explained in other limitations, including 1[a], that a POSITA would have recognized the DIDS director as a "trust orchestration server." Mukherjee discloses identifying threats to the network including targets such as the systems and applications running on the system.

> ***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. 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. Misuses represent out-of-policy behavior in which the state of the machine is not affected. Suspicious acts are events which, while not a violation of policy, are of interest to an IDS. For example, commands which provide information about the state of the system may be suspicious. ***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 non-privileged 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).

224. A POSITA would have recognized this analysis of "threats" and "abuses", especially "attacks" representing maliciously changed machine state, as an "integrity profile" for the system.

225. Miliefsky also discloses "generating… an integrity profile of the system." As Miliefsky explains, "***The method defines a true risk profile for the computer-based network environment***, and uses the knowledge of external and internal CVEs as well as how to manage and remediate against these CVEs, to provide more robust and proactive security." (Ex. 1006 at [0045]). Miliefsky also teaches displaying the risk profiles of assets (e.g., equipment) on administrative dashboards. "The administrative dashboard provides access to and control of advanced alerting operations such as alert through e-mail or pager module on regulatory compliance reporting and related network-based asset risk profile." (*Id.* at [0070]). A POSITA would have recognized a risk profile, generated from vulnerability assessments, to be an "integrity profile" of the system.

226. A POSITA would have found Mukherjee's and Miliefsky's integrity profiles to be compatible, as they teach similar concepts. In particular, Miliefsky's dashboard "provides access

to and control of advanced alerting operations," as cited above. Mukherjee's disclosed UI provides control of the "entire system."

> 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).

227. Thus, I conclude that Mukherjee and Miliefsky each disclose limitation 1[h].

228. Thus, I conclude that Mukherjee in view of Miliefsky renders Claim 1 obvious to a POSITA.

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

229. Claim 2 reads:

> *The method of claim 1, wherein the third party network endpoint assessments are received from an endpoint assessment service.*

230. I have already explained that Mukherjee, or Mukherjee in view of Miliefsky, discloses third-party network endpoint assessments with respect to limitation 1[e]. I incorporate my analysis of limitation 1[e] into this section by reference.

231. As I explained with reference to limitation 1[e], Miliefsky also notes that compatible vulnerability assessment systems that use OVAL can provide data to Miliefsky's system. Such systems can include a "Web site", which a POSITA would have understood to include web sites hosted by third parties:

> ***An "OVAL-compatible" tool, service, Web site, database, or advisory/alert*** uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state. ***An "OVAL-ID compatible" tool, Web site, database, archive, or security advisory*** includes OVAL-ids as part of the information it conveys about a security issue, and provides for searching by OVAL ID with potential linkage back to the source definition of the OVAL-ID.
> (Ex. 1006 at [0104], emphasis added).

232. A POSITA would have understood that assessment systems hosted by a third-party website would represent "an endpoint assessment service." (*See* my analysis of limitation 1[e]).

233. Thus, I conclude that Mukherjee in view of Miliefsky discloses Claim 2.

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

234. Claim 3 of the '616 patent is rendered obvious to a POSITA by Mukherjee in view of Miliefsky. This claim, which depends on Claim 1, reads:

> *The method of claim 2, further comprising: receiving, by the trust orchestration server, integrity measurement and verification reports from the endpoint assessment service.*

235. The specification of the '616 patent explains that "network based integrity measurement and verification services scan and audit the soft-core that comprises business critical systems, services and high value data silos." (Ex. 1001 at 2:24-34). It describes these as being produced either by "scans performed by a system configuration sensor 113" or "from an endpoint assessment service 820." (Ex. 1001 at 8:23-32 and 31:1-3).

236. The specification also identifies integrity measurement and verification reports as something similar to, or an alternative to, endpoint assessments: "… the trust broker 407 receives and analyzes network endpoint assessments (or integrity measurement and verification reports) 821 from an endpoint assessment service (or collaboration services) 820." (Ex. 1001 at 32:63-67). However, I understand that Claim 3, which depends from Claim 2, which in turn depends from Claim 1, requires both "receiving, by the trust orchestration server, third party network endpoint assessments" (limitation 1[e]) "from an endpoint assessment service" (Claim 2) and "receiving, by the trust orchestration server, integrity measurement and verification reports from the endpoint assessment service" (Claim 3).

237. In addition to producing endpoint assessments from third-party endpoint assessment services, Miliefsky also produces audit and compliance information that a POSITA would recognize as "integrity measurement and verification report[ing]." Miliefsky discloses that OVAL compatible systems incorporate XML distribution feeds from providers that enable auditing according to a specification:

> ***When talking about OVAL compatibility*** it is necessary to consider each of these schemas and how they will be used. For each schema there is a notion of "producers" and "consumers." Typically, ***a Feed Provider, today, provides news and related information as an XML Feed***, through various Aggregators to end Users (Consumers)…
>
> Feed consumers—there will be two feed consumers—people and INFOSEC countermeasures… ***the countermeasures will be able to use the feed to dynamically reconfigure themselves based on Global and local security threats as well as the internal vulnerabilities or weaknesses found in the internal assets*** through a real-time CVE® differential analysis performed by the preferred embodiment system.
>
> ***Regulations such as Sarbanes-Oxley, GLBA, HIPAA, and others are only the beginning. The possibility of an upcoming cybersecurity audit mandated by the SEC looms large.*** The need for proactive and regular external IT security audits as well as internal controls has led to the need of real-time feeds—an RSS feed-based INFOSEC solution. As a result, networks will be more secure, experience more uptime and outdated INFOSEC countermeasure equipment—Firewalls/VPNs, IDS, IPS, Antivirus, etc. will be able to perform in an optimized fashion, taking a more holistic view of network security, in real-time, based upon the new and critical RSS feed provided by the preferred embodiment.
> (Ex. 1006 at [0107]-[0110], emphasis added).

238. Miliefsky further explains the design of the system includes handling audit and compliance:

> These INFOSEC engine plugins may include new interfaces to various countermeasures (i.e. Firewalls, VPNs, IDS and IPS), enhanced or new CVE auditing functionality, ***enhanced or new regulatory compliance reporting***, enhanced or new policy building tools, ***enhanced auditing capabilities*** such as rogue wireless device detection, mobile device detection, updated database tables, updated GUI features and other 'packaged' enhancements to maintain currency of the system.

(Ex. 1006 at [0067], emphasis added).

239. I have already explained that a POSITA would have been motivated to combine Mukherjee and Miliefsky with respect to limitation 1[e] and Claim 2. I incorporate my analysis of this motivation to combine into this section by reference.

240. Thus, Mukherjee in view of Miliefsky renders Claim 3 obvious to a POSITA.

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

241. Claim 4 of the ’616 patent is rendered obvious to a POSITA by Mukherjee in view of Miliefsky. This claim, which depends on Claim 1, reads:

> *The method of claim 1, wherein the third party network endpoint assessments are received from one or more collaboration services.*

242. Claim 4 is identical to Claim 2 except that “endpoint assessment service” is replaced with “collaboration services.” Within the specification of the ’616 patent, “endpoint assessment services” and “collaboration services” can perform identical functions: “In step 2103, the trust broker 407 receives and analyzes network endpoint assessments (or integrity measurement and verification reports) 821 from an endpoint assessment service (or collaboration services) 820 and generate temporal events 409.” (Ex. 1001 at 32:63-67).

243. I have already explained with respect to Claim 2 that Miliefsky discloses OVAL-compatible tools, databases, and Web sites and a POSITA would have recognized third-party “Web sites” as the recited "endpoint assessment services.” A POSITA would have recognized that the “tools, databases, and Web sites” could also be “collaboration services.”

244. I have already explained that a POSITA would have been motivated to combine Mukherjee and Miliefsky with respect to Claim 2. I incorporate my analysis of this motivation to combine into this section by reference.

245. Thus, Mukherjee in view of Miliefsky renders Claim 4 obvious to a POSITA.

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

246. Each of the limitations of Claim 5 of the ’616 patent, which depends from Claim 4, is rendered obvious to a POSITA by Mukherjee in view of Miliefsky as described below.

247. Claim 5 reads:

> *The method of claim 4, further comprising: receiving, by the trust orchestration server, integrity measurement and verification reports from the one or more collaboration services.*

248. Claim 5 is identical to Claim 3 except that “endpoint assessment service” is replaced with “collaboration services.” Within the specification of the ’616 patent, “endpoint assessment services” and “collaboration services” can perform identical functions: “In step 2103, the trust broker 407 receives and analyzes network endpoint assessments (or integrity measurement and verification reports) 821 from an endpoint assessment service (or collaboration services) 820 and generate temporal events 409.” (Ex. 1001 at 32:63-67).

249. I have already explained with respect to Claim 2 (from which Claim 3 depends) that Miliefsky discloses OVAL compatible tools, databases, and Web sites and a POSITA would have recognized third-party “Web sites” as the recited "endpoint assessment services.” A POSITA would have recognized that the “tools, databases, and Web sites” could also be “collaboration services.”

250. I have already explained with respect to Claim 3 that Miliefsky teaches “integrity measurement and verification reports”. In addition to producing endpoint assessments from third-party endpoint assessment services (relevant to Claim 3), Miliefsky also produces audit and compliance information that a POSITA would recognize as “integrity measurement and verification report[ing].” Miliefsky discloses that OVAL compatible systems incorporate XML distribution feeds from providers that enable auditing according to a specification:

> ***When talking about OVAL compatibility*** it is necessary to consider each of these schemas and how they will be used. For each schema there is a notion of "producers" and "consumers." Typically, ***a Feed Provider, today, provides news and related information as an XML Feed***, through various Aggregators to end Users (Consumers)…
>
> Feed consumers—there will be two feed consumers—people and INFOSEC countermeasures… ***the countermeasures will be able to use the feed to dynamically reconfigure themselves based on Global and local security threats as well as the internal vulnerabilities or weaknesses found in the internal assets*** through a real-time CVE® differential analysis performed by the preferred embodiment system.
>
> ***Regulations such as Sarbanes-Oxley, GLBA, HIPAA, and others are only the beginning. The possibility of an upcoming cybersecurity audit mandated by the SEC looms large.*** The need for proactive and regular external IT security audits as well as internal controls has led to the need of real-time feeds—an RSS feed-based INFOSEC solution. As a result, networks will be more secure, experience more uptime and outdated INFOSEC countermeasure equipment—Firewalls/VPNs, IDS, IPS, Antivirus, etc. will be able to perform in an optimized fashion, taking a more holistic view of network security, in real-time, based upon the new and critical RSS feed provided by the preferred embodiment.
> (Ex. 1006 at [0107]-[0110], emphasis added).

251. Miliefsky further explains the design of the system includes handling audit and compliance:

> These INFOSEC engine plugins may include new interfaces to various countermeasures (i.e. Firewalls, VPNs, IDS and IPS), enhanced or new CVE auditing functionality, ***enhanced or new regulatory compliance reporting***, enhanced or new policy building tools, ***enhanced auditing capabilities*** such as rogue wireless device detection, mobile device detection, updated database tables, updated GUI features and other 'packaged' enhancements to maintain currency of the system.
> (Ex. 1006 at [0067], emphasis added).

252. I have already explained that a POSITA would have been motivated to combine Mukherjee and Miliefsky with respect to Claim 3. I incorporate my analysis of this motivation to combine into this section by reference.

253. Thus, I conclude that Mukherjee in view of Miliefsky renders Claim 5 obvious to a POSITA.

G. Claim 6 is rendered obvious by Mukherjee in view of Miliefsky

254. Claim 6 of the ’616 patent, which depends on Claim 1, is rendered obvious to a POSITA by Mukherjee in view of Miliefsky as described below.

1. Limitation 6[a]

255. Limitation 6[a] of the ’616 reads:

> *The method of claim 1, wherein the endpoint events of the monitored device are received by a system event correlator of the trust orchestration server from endpoint trust agents, and*

256. I have already explained with respect to limitation 1[a] that Mukherjee discloses a “system event correlator” in the form of the DIDS director. I incorporate my analysis of limitation 1[a] into this section by reference.

257. Thus, Mukherjee discloses limitation 6[a].

2. Limitation 6[b]

258. Limitation 6[b] of the ’616 patent reads:

> *the temporal events are generated by a trust broker of the trust orchestration server and are received by the system event correlator from the trust broker.*

259. Mukherjee in view of Miliefsky discloses limitation 6[b].

260. As explained with respect to limitation 1[f], Miliefsky discloses “generating… temporal events.” For example, Miliefsky discloses generating lists of equipment assets and their configuration.

> ***As part of the security method, in a first step the network is scanned and/or probed for any and all attached equipment and related assets, herein referred to as "network-based" assets***. The method will dynamically detect and map changes to LAN and WAN connected equipment including searching for equipment which may be deemed as rogue and creating a network-based assets list, wherein the list contains information as to the location of the network-based assets.
>
> ***The list may contain information as to the Internet Protocol (IP) address of said network-based assets, and the list may contain information as to the open ports of said network-based assets and related application, session, transport, sockets and other internet protocol (IP) related information. The list may contain other information such as the Media Access Control (MAC) address of said network-based assets, whether the connection is Wired or Wireless of said network-based assets and other information about the structure of the network and its component devices***. (*Id.* at [0040]-[0041], emphasis added).

261. As discussed with reference to limitation 1[f] above, the specification for the '616 patent specifically calls out OVAL as a format for storing temporal event data, a format that Miliefsky also uses. In particular, the '616 specification explains that temporal events can be "expressed as assertions about operational parameters (e.g., vulnerabilities, compliance, patch level, etc.) based on enterprise policies established for a baseline configuration." (Ex. 1001 at 11:52-55). It states further,

> With continued reference to FIG. 4, the temporal events 409 ***can be an annotation (e.g., an XML representation) of the configuration state of the system*** (including device, application and package) and include trust and severity scores as generated by third party security vendors based on integrity measurement and verification, in accordance with standards established by organizations… ***temporal events can be assertions about operational parameters expressed in the Open Vulnerability and Assessment Language (OVAL) to convey system details***…

262. (Ex. 1001 at 11:59-12:16, emphasis added).

263. Miliefsky explains that assessment data is collected using a standard language called OVAL:

> OVAL is the Open Vulnerability Assessment Language. It is funded by the U.S. Department of Homeland Security (DHS) and in summary the XML,

> machine readable format for the Common Vulnerabilities and Exposures (CVE®) standard. OVAL is an international, information security community baseline standard for how to check for the presence of vulnerabilities and configuration issues on computer systems. ***OVAL standardizes the three main steps of the process with an OVAL System Characteristics Schema for collecting configuration data from systems for testing; OVAL Definitions to test for the presence of specific vulnerabilities, configuration issues, and/or patches; and an OVAL Results Schema for reporting the results from the evaluated systems.*** The tests are standardized, machine-readable XML Vulnerability Definitions, Compliance Definitions, and Patch Definitions. OVAL's schemas and definitions are all free to download, use, reference, and implement. An "OVAL-compatible" tool, service, Web site, database, or advisory/alert uses the Open Vulnerability and Assessment Language (OVAL), as appropriate, for communicating details of vulnerabilities, patches, security configuration settings, or machine state.
> (Ex. 1006 at [0103]-[0104], emphasis added).

264. And, as explained with reference to limitation 1[e], Miliefsky then identifies that the database that holds results for this system can include external, third-party assessments in addition to its own.

> Database Correlation and Warehousing Engine: ***This engine gathers data from processes and results throughout the product as well as from internal/external resources, including but not limited to the Update Servers, Countermeasure appliances, Data feeds, other devices of the same nature as this patent describes and any related third party sources***. The engine uses data warehouse methodologies to store this data. The engine also provides a means of querying the database and warehouse information either through automated methods or through on-demand user interfaces.
> (Ex. 1006 at [0093], emphasis added).

265. Thus, the "engine" used in the "Database Correlation and Warehousing Engine" is a "trust broker," because it receives data from many sources and makes them available through "on-demand user interfaces."

266. Thus, Mukherjee in view of Miliefsky discloses limitation 6[b].

267. Thus, I conclude that Mukherjee in view of Miliefsky renders Claim 6 obvious to a POSITA.

X. GROUND 2: CLAIM 7 IS RENDERED OBVIOUS BY MUKHERJEE IN VIEW OF MILIEFSKY IN FURTHER VIEW OF GUO

268. Claim 7 of the '616 patent is rendered obvious to a POSITA by Mukherjee in view of Miliefsky in further view of Guo, as explained in the sections below. I also explain that it would have been obvious for a POSITA to combine Mukherjee and Miliefsky and Guo.

A. Motivation to Combine Mukherjee and Miliefsky and Guo

269. I have already explained that it would have been obvious to a POSITA to combine Mukherjee and Miliefsky, and that the POSITA would have been motivated to do so in Section IX.A. I incorporate this analysis into this section by reference.

270. Furthermore, it would have been obvious for a POSITA to combine Mukherjee and Miliefsky with Guo. As I have already explained, Mukherjee teaches that the DIDS system is meant to be used in concert with other systems specifically to obtain real-time defenses.

> ***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).

271. A POSITA would have understood that Guo teaches "real-time" defense. Guo explains that "[t]o detect malicious code that is dynamically created (i.e., ***created at runtime***), e.g., by a packer, antivirus software may require ***dynamically created code to be scanned before it can be executed by a host computer system***." (Ex. 1011 at 2:22-25, emphasis added). Thus, a POSITA would have recognized that Guo provides a real-time anti-malware defense to enable "actively respond[ing] to attacks against the system in real-time," as Mukherjee teaches.

272. Miliefsky is even more explicit in disclosing a security appliance that incorporates manifold security operations, each of which is provided through a plugin. One such plugin is an "anti-virus" (AV) plugin. "The attached figures illustrate a proactive network security system to protect against hackers, which comprises a[n]… INFOSEC engine PLUGINs and communications interface possibly including but not limited to one or more of the following: … AntiVirus Servers (AVS)…" (Ex. 1006 at [0046]). Guo discloses an enhanced AV system: "… the system described here functions by accurately catching this moment and invoking the AV scanner at that point so that existing AV scanning technology can be applied to [packed data] as effectively as to [unpacked data]…" (Ex. 1011 at 4:10-13).

273. Thus, a POSITA would have known that both Mukherjee and Miliefsky are explicitly expected to combine with AV systems such as Guo's. The systems are compatible and expected to work together.

274. A POSITA would have been motivated to use Guo's AV system because Guo's system is designed to catch evasive malware. As Guo explains, "malware writers can easily generate a large number of variants from the same malware." (Ex. 1011 at 50-51). As opposed to manual reverse engineering, which is slow and expensive, Guo's general and automated approach "… relates generally to antivirus software for detecting malicious code, and in particular to ***preventing packers from evading the scanning of their dynamically created malicious code***." (*Id.* at 1:15-18, emphasis added). Guo further explains the advantages by stating that, "the packing manager 120 is generic because the way it tracks the code pages of the original programs ***does not require any packer-specific knowledge***." (*Id.* at 5:18-20, emphasis added). Moreover, "it is efficient because page status tracking can be done transparently by virtual memory hardware, and thus does not incur any software overhead." (*Id.* at 5:22-24).

275. A POSITA also would have been motivated to use Guo because its packing detection system can detect multiple packing, wherein the data is packed multiple times, potentially by different packers:

> The packing manager system 120 can deal with multiple-layer packed binaries equally effectively without any modification. Since each layer of packer exhibits the same behavior, executing dynamically generated code, each layer will be unpacked in the reverse order of their packing sequence. If the latest unpacking round is not the last round, its output is an intermediate packed binary that needs to be unpacked. In this case, the program resumes and the packing manager 120 continues to track pages that are dynamically generated and executed.
> (Ex. 1011 at 5:32-41).

276. Finally, as noted in Section V.C., the existence of obfuscation alone is not sufficient to determine if a program is malware. Rather, it is necessary to restore the original code and determine if the original is malicious. A POSITA would have been motivated to use Guo because this is exactly the problem that it solves: automatically detecting and unpacking any number of layers of packing in order to analyze the original.

277. Thus, a POSITA would have been motivated to combine Guo with Mukherjee and Miliefsky, and would have had a high expectation of success with minimal experimentation.

B. Claim 7 is rendered obvious by Mukherjee in view of Miliefsky in further view of Guo

278. Claim 7 of the ’616 patent is rendered obvious to a POSITA by Mukherjee in view of Miliefsky in further view of Guo. This claim reads:

> *The method of claim 1, further comprising: performing, by a malware analyzer, detection of code obfuscation and evasion techniques in reverse engineered code of the applications, the code obfuscation and evasion techniques including at least one of: multiple packing, anti-debug, anti-trace, anti-memory, and anti-emulation.*

279. Claim 7 depends from Claim 1. I have already explained in Section IX.B. that Mukherjee in view of Miliefsky renders Claim 1 obvious to a POSITA. I incorporate this analysis into this section by reference.

280. Guo discloses the additional requirements of Claim 7. Guo discloses "performing, by a malware analyzer, detection of code obfuscation and evasion techniques in reverse engineered code of the applications, the code obfuscation and evasion techniques including… multiple packing."

281. Guo discloses a malware analyzer. Guo describes a "malicious code detection module 340." (Ex. 1011 6:56-57). Guo explains that this module works with another component called the "packing manager 120", as also illustrated in Figure 3, reproduced hereinbelow.

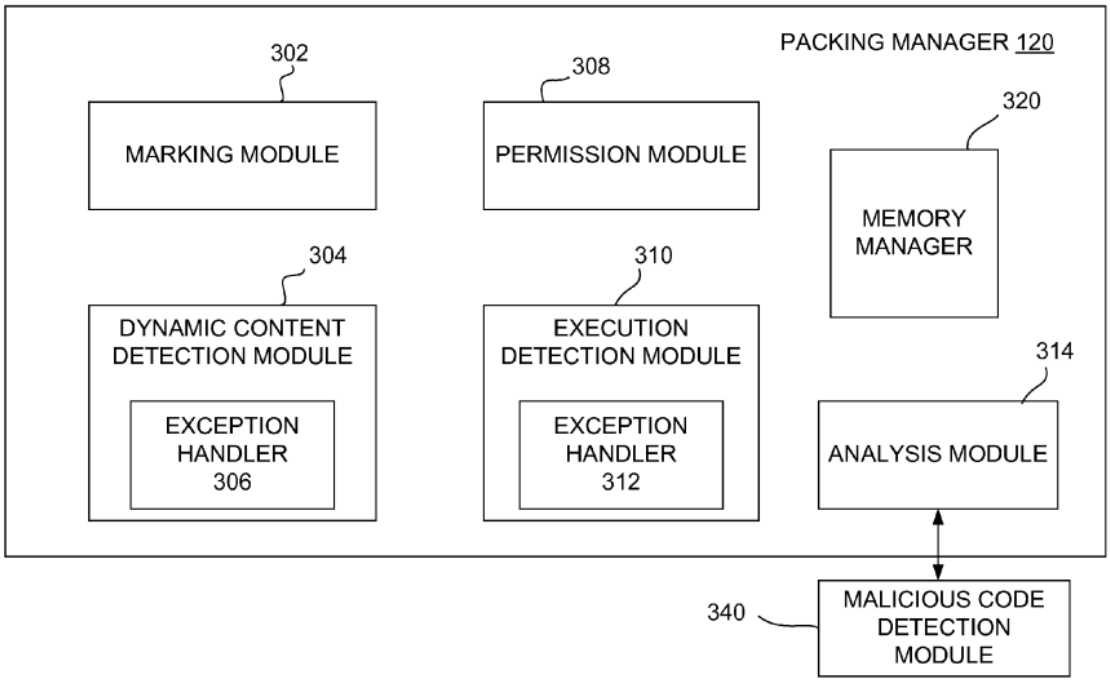


FIGURE 3

(Ex. 1011 at Figure 3).

282. Guo explains the relationship between malicious code detection module 340 and packing manager 120 further:

> The malicious code detection module 340 can be an AV scanner for scanning the program for the presence of malicious code. In FIG. 3, the malicious code detection module 340 is shown as separate from the packing manager 120, but it could also be included partially or entirely within the packing manager 120.
> (Ex. 1011 at 9:31-36).

283. Guo discloses that the packing manager 120 (which uses or is combined with the malicious code detection module 340) detects code obfuscation and evasion techniques in which the code obfuscation and evasion techniques include multiple packers. Guo explains:

> The packing manager system 120 can deal with ***multiple-layer packed binaries equally effectively without any modification***. Since each layer of packer exhibits the same behavior, executing dynamically generated code, each layer will be unpacked in the reverse order of their packing sequence. If the latest unpacking round is not the last round, its output is an intermediate packed binary that needs to be unpacked. In this case, the program resumes and the packing manager 120 continues to track pages that are dynamically generated and executed.
> (Ex. 1011 at 5:32-41, emphasis added).

284. Guo is explicit that "multiple-layer packed binaries" is "multiple packing." Guo explains that "[m]alware writers… can apply packing multiple times to a malware O and ***generate such variants as P1(P2(O)), P2(P1(O)), and so on***. By applying multi-layer packing, even with a small number of packers, malware writers can easily generate a large number of variants from the same malware." (Ex. 1011 at 3:44-51, emphasis added). Guo also states that this is a form of evasion: "…malware writers can also apply different packers in different combinations to create a large number of variants of existing malware ***that can easily evade signature-based AV scanners***." (*Id.* at 1:39-42, emphasis added). Guo also acknowledges that packing is a form of obfuscation: "The term, 'unpacking,' as used here includes unpacking, unencrypting, descrambling, ***de-obfuscating***, etc." (*Id.* at 5:50-52, emphasis added). Thus, Guo discloses "detection of code obfuscation and evasion techniques… including… multiple packing."

285. Finally, in my opinion, the plain and ordinary meaning for "detection… in reverse engineered code of the applications" is "detecting characteristics or metadata about the applications as a result of performing static or dynamic analysis of the binary application." A POSITA would have understood Guo's techniques to detect characteristics about the applications as a result of performing dynamic analysis. As discussed in Section VIII.C., Guo's approach is to process the application in real-time. Guo explains that "[t]o detect malicious code that is dynamically created (i.e., ***created at runtime***), e.g., by a packer, antivirus software may require ***dynamically created code to be scanned before it can be executed by a host computer system***." (Ex. 1011 at 2:22-25, emphasis added). Thus, Guo's approach is "dynamic analysis."

286. Moreover, Guo detects characteristics about the application as a result of this analysis. Specifically, Guo detects memory pages that are both created (written) and executed. As explained in more detail in Section VIII.C., Guo details this processing:

> Regardless of the packer used, the unpacker routine in a packed binary at some point creates the original program being packed at run time and transfers control to it. So, during the execution of a packed binary, the code pages associated with the original program must be dynamically generated and later on executed. By catching control transfers to dynamically written pages, the system described here can catch the moment when the original program is restored.
> (Ex. 1011 at 4:26-34).

287. Guo explains further:

> More specifically, before an executable binary is executed in this embodiment, the packing manager system 120 marks its memory pages as non-writable but executable… ***When the unpacker starts restoring the original program, it will write to non-writable pages*** (marked NW by the packing manager) and will cause a write access violation exception for the first write to each such page… The virtual memory hardware automatically detects the write exception and the operating system delivers this exception to the user process. The packing manager 120 intercepts and handles each such exception by marking the associated page as writable but non-executable and resuming the execution so that the unpacker can continue restoring the original program…

> ***After the unpacker fully restores the original program, it transfers control to the entry-point (the first instruction) of the original program. The virtual memory hardware detects an execution exception because that page is writable but not executable.*** The operating system again delivers this exception to the user process. The packing manager 120 catches this exception and calls AV scanner to scan the address space snapshot at this point. For example, the system can dump the process's address space image for an AV scanner to scan. If the AV scanner detects malware, the packing manager 120 terminates the process.
> (Ex. 1011 at 4:51-5:9, emphasis added).

288. Thus, Guo teaches identifying obfuscation techniques through dynamic analysis. As also noted in Section X.A., Guo teaches that its disclosures are an alternative to ***manual*** reverse engineering. "Traditionally, AV companies have attempted to manage the packer problem by ***manually reverse-engineering*** popular packers and creating unpackers for them… [but] reverse-engineering each of these different packers is an impossible proposition because it is a slow and expensive process." (Ex. 1011 at 1:43-52, emphasis added). A POSITA would have understood Guo's disclosures, which automatically extract the original malware code from evasive packing to be "reverse engineering" the code, using the plain and ordinary meaning of the term as it would have been understood by a POSITA.

289. Thus, Guo discloses the additional requirements of Claim 7.

290. A POSITA would have been motivated to combine Guo with Mukherjee and Miliefsky, as I explained in Section X.A. I incorporate this analysis into this section by reference.

291. Thus, Mukherjee in view of Miliefsky in further view of Guo renders Claim 7 obvious to a POSITA.

XI. THERE ARE NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

292. I am not aware of any objective indicia of non-obviousness related specifically to the '616 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 ’616 patent.

XII. CONCLUSION

293. 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 5 April , 2023

By: _____
Seth James Nielson