IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

_____

BEFORE THE PATENT TRIAL AND APPEAL BOARD

_____

TREND MICRO, INC.
Petitioner,

v.

WEBROOT, INC.
Patent Owner.

_____

IPR2023-01390

U.S. Patent No. 8,201,243

DECLARATION OF SETH JAMES NIELSON

TABLE OF CONTENTS

I. INTRODUCTION 1

II. QUALIFICATIONS 1

III. MATERIALS CONSIDERED 12

IV. LEGAL STANDARDS 13

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

A. Log aggregation 16

B. System calls, hooking, and the Windows API 18

C. Processes and files in Windows monitoring 20

D. Behavior monitoring techniques for anti-malware 22

VI. OVERVIEW OF THE CHALLENGED PATENT 27

A. THE ’243 PATENT SPECIFICATION AND CLAIMS 27

B. PROSECUTION HISTORY 29

VII. LEVEL OF SKILL IN THE ART 31

VIII. Overview of the prior art 31

A. OVERVIEW OF ABAD 31

B. OVERVIEW OF PICCARD 35

C. OVERVIEW OF KILPATRICK 38

IX. GROUND 1: Claims 1-14 are rendered obvious by Abad in view of Piccard 41

A. Motivation to Combine Abad and Piccard 41

B. Claim 1 is rendered obvious by Abad in view of Piccard 45

1. Limitation 1[a] 45

2. Limitation 1[b] 47

3. Limitation 1[c] 48

4. Limitation 1[d] 51

5. Limitation 1[e] 52

6. Limitation 1[f] 55

C. Claim 2 is rendered obvious by Abad in view of Piccard 58

D. Claim 3 is rendered obvious by Abad in view of Piccard 60

E. Claim 4 is rendered obvious by Abad in view of Piccard 62

F. Claim 5 is rendered obvious by Abad in view of Piccard 64

G. Claim 6 is rendered obvious by Abad in view of Piccard 66

1. Limitation 6[a] 66

2. Limitation 6[b] 68

3. Limitation 6[c] 68

4. Limitation 6[d] 71

5. Limitation 6[e] 74

H. Claim 7 is rendered obvious by Abad in view of Piccard 77

I. Claim 8 is rendered obvious by Abad in view of Piccard 78

J. Claim 9 is rendered obvious by Abad in view of Piccard 79

K. Claim 10 is rendered obvious by Abad in view of Piccard 80

L. Claim 11 is rendered obvious by Abad in view of Piccard 82

M. Claim 12 is rendered obvious by Abad in view of Piccard 84

1. Limitation 12[a] 84

2. Limitation 12[b]-12[f] 86

N. Claim 13 is rendered obvious by Abad in view of Piccard 88

O. Claim 14 is rendered obvious by Abad in view of Piccard 88

X. GROUND 2: Claims 1-14 are rendered obvious by Kilpatrick in view of Piccard 89

A. Motivation to Combine Kilpatrick and Piccard 89

B. Claim 1 is rendered obvious by Kilpatrick in view of Piccard 91

1. Limitation 1[a] 91

2. Limitation 1[b] 93

3. Limitation 1[c] 95

4. Limitation 1[d] 97

5. Limitation 1[e] 99

6. Limitation 1[f] 100

C. Claim 2 is rendered obvious by Kilpatrick in view of Piccard 102

D. Claim 3 is rendered obvious by Kilpatrick in view of Piccard 104

E. Claim 4 is rendered obvious by Kilpatrick in view of Piccard 105

F. Claim 5 is rendered obvious by Kilpatrick in view of Piccard 107

G. Claim 6 is rendered obvious by Kilpatrick in view of Piccard 108

1. Limitation 6[a] 108

2. Limitation 6[b] 111

3. Limitation 6[c] 112

4. Limitation 6[d] 115

5. Limitation 6[e] 117

H. Claim 7 is rendered obvious by Kilpatrick in view of Piccard 119

I. Claim 8 is rendered obvious by Kilpatrick in view of Piccard 120

J. Claim 9 is rendered obvious by Kilpatrick in view of Piccard 121

K. Claim 10 is rendered obvious by Kilpatrick in view of Piccard 122

L. Claim 11 is rendered obvious by Kilpatrick in view of Piccard 124

M. Claim 12 is rendered obvious by Kilpatrick in view of Piccard 125

1. Limitation 12[a] 125

2. Limitation 12[b]-12[f] 128

N. Claim 13 is rendered obvious by Kilpatrick in view of Piccard 130

O. Claim 14 is rendered obvious by Kilpatrick in view of Piccard 130

XI. There are no objective indicia of nonobviousness 130

XII. Conclusion 131

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

I. INTRODUCTION

2. I have been retained by counsel for Trend Micro, Inc. ("Trend Micro" or "Petitioner") as a technical expert in connection with the proceeding identified above. I submit this Declaration in support of Petitioner Trend Micro's Petition for Inter Partes Review (the "Petition") of United States Patent No. 8,201,243 ("the '243 patent") against Webroot, Inc. ("Webroot" or "Patent Owner"). All "Ex. 10XX" cites herein are to the Exhibits to the Petitions. All "Ex. _" cites followed by a letter are to exhibits to this Declaration. 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 '243 patent are obvious in light of the prior art discussed below. The '243 patent is directed to general application of known machine learning techniques to the identification of malware.

4. In summary, my conclusion is that the claims of the '243 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 1-14 are rendered obvious by the combination of Abad and Piccard.

b. Claims 1-14 are rendered obvious by the combination of Kilpatrick and Piccard.

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. A copy of my current C.V. is attached hereto as **Exhibit 1004**. My C.V. 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 '243 patent and its file history, the other documents and references as cited herein, and general technical references. I understand that Counsel for Petitioner have submitted a Petition that reference by exhibit number. For simplicity and convenience, I have adopted the same exhibit numbering.

50. The exhibits are:

- Ex. 1001: U.S. Patent No. 8,201,243 to Matthew L. Boney. ("the '243 patent")
- Ex. 1002: Prosecution history of the '243 patent
- Ex. 1003: Expert Declaration of Dr. Seth James Nielson
- Ex. 1004: Curriculum Vitae of Dr. Seth James Nielson
- Ex. 1005: Webroot, Inc. and Open Text, Inc. v. Trend Micro Inc., Case No. 6:22-cv-00243-ADA-DTG, Claim Construction Order, Dkt. 391 (W.D. Tex., July 25, 2023).
- Ex 1006: Cristina Abad, *Log Correlation for Intrusion Detection: A Proof of Concept*, 19th Annual Computer Security Applications Conference (2003). ("Abad")
- Ex. 1007: U.S. Patent Publication No. 2007/0006310 to Piccard. ("Piccard")
- Ex. 1008: U.S. Patent No. 6,735,703 to Kilpatrick et al. ("Kilpatrick")

- Ex. 1009: Ed Skoudis, *Malware: Fighting Malicious Code*, Pearson (Nov. 2003).
- Ex. 1010: Robert Rinnan, *Benefits of Centralized Log File Correlation*, Gjøvik University College (2005).
- Ex. 1011: Platform SDK: Windows Management Instrumentation, Win32_Process, Win32_Process class [WMI] (archive.org) (Jul. 17, 2004).
- Ex. 1012: Matthew M. Lavy, *Windows Management Instrumentation (WMI)*, Sams (Oct. 2001).
- Ex. 1013: Yariv Kaplan, *API Spying Techniques for Windows 9x, NT and 2000*, API Spying Techniques for Windows 9x, NT and 2000 (archive.org) (Feb. 10, 2003).
- Ex. 1014: Stephanie Forrest, *Computer Immunolgy*, Communications of the ACM 41(1), (Oct. 1997).

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 that in a parallel district court litigation the owner of the ’243 patent is arguing that the ’243 patent claims should be afforded their plain and ordinary meaning. I have also been requested to adopt patent owner’s position on claim construction for the purposes of rendering my opinions set forth in this declaration.

53. I understand that a patent may include independent claims and dependent claims. An independent claim stands by itself and 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 claims include 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; I 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, but that sometimes common sense as would have been known to a POSITA can provide a reason to link 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 of endeavor 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 of endeavor 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 of endeavor 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 U.S. patent claims may be granted an earlier effective filing date if the patent owner properly claimed priority to an earlier patent application that fully supports the claims.

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 '243 patent that would have been known to a POSITA as of the filing date of the '243 patent, which I understand to be April 20, 2006. Any discussion of the background technologies in this section is written to be as understood by a POSITA of that date.

A. Log aggregation

63. It was well understood by practitioners of the art at the time of the alleged invention that correlating logs from disparate sources improved intrusion detection.

> The problem faced today by system administrators are the correlation of Firewall logs, intrusion detection system event logs, operating system event logs, mail system logs, database logs, web server logs, antivirus logs, router/switch logs etc. [10] states that potentially, all these logs can identify a threat, and may receive hundreds or thousands of entries a day. The examination of these logs is often performed by staff short of time and knowledge, and for a company; resources may be a limitation. ***Despite the resource limitations we want log file correlation because it improves intrusion detection*** [12].
> (Ex. 1010, p. 15, emphasis added).

64. Many commercial products were available that had this capability, as well as related capabilities necessary to make such correlation function well, such as aggregation:

> Reading some of the whitepapers and briefings on products web-pages provides some insight into their architecture, and motivation. There seem to be differences among the products, but ***they all seem to follow a layout consisting of normalization, aggregation, correlation, and visualization***. As an example the netForensics product [17] automatically aggregates events from thousands of disparate sources into a single data format, normalizes the events into 100 event types across nine categories, making it possible to automatically identify events using correlation, generate reports, and perform analysis.
> (*Id.*, p. 15, emphasis added).

65. Rinnan gives an overview of "the approaches found in the literature", several of which are centralization, consolidation, and aggregation. (*Id.*, pp. 16-17).

> The centralization process may be defined as gathering the log files/entries in one server.
> (*Id.*, p. 16).

> Consolidation is the process that takes data from different systems and entities, and possibly disparate formats, and combines and aggregates that information to create a unified view [42].
> (*Id.*, pp. 16-17).

> Data aggregation [2, 23] organizes normalized data by category, for instance IT systems, applications, etc. [4] states that it is necessary to eliminate redundant or duplicate event data from the security event data stream. This includes refining and optimizing the amount of information that is presented to security analysts. … Grouping similar events together and providing answers to how many times an attack happened over a certain time period, enables security experts to group and count these events quickly by port, source IP, target IP, business asset (e.g. Web servers or mail servers.) or other parameters [11]. [19] provides filtering examples such as protocol type, time, IP, and MAC Address.
> (*Id.*, p. 17).

66. Thus, a POSITA would have understood the value of collecting various types of log information to a centralized location and aggregating it. Furthermore, a POSITA would have

understood the value of storing such information in a common file or database, such as described here:

> [4] defines normalization as the process of ***gathering individual security device data, and putting them into context that is easier to understand***, and claims that this enables us to map different messages about the same security event to a common alarm ID. [4] claims that normalization alone is a tremendous asset to security teams. Some commercial products use XML to structure the entries and a paper from GuardedNet [5] claims that ***normalization is a process that breaks an event into its component parts and places them into individual database fields***. The use of a database is also supported by Johan Beckers and Jean Paul Ballerini [11].
> (*Id.*, p. 16, emphasis added).

B. System calls, hooking, and the Windows API

67. System calls are a mechanism by which user-mode processes can access system resources (e.g., the file system) to which the operating system wishes to control access. In Windows, such kernel-level functions are not exposed directly to user-mode programs but are instead wrapped by the Windows API:

> If you have ever examined ntdll.dll with QuickView, you might have noticed that it exports a set of functions that begin with the Nt prefix. These functions are actually small stubs of code that pass control to the Windows NT kernel (NTOSKRNL) using interrupt 2E. Many of the functions exported from kernel32.dll are nothing more than control transfer routines to the stubs located in ntdll. For example, when a Windows application issues a call to CreateFile located in kernel32.dll, the call is redirected to NtCreateFile, which passes it on to NT's kernel for further processing. The special design of this mechanism allows a device driver to hook these interfaces, thus providing a way for monitoring activities performed by Windows NT/2000 applications.
> (Ex. 1013, p. 6).

68. A POSITA would have understood that in Windows, invoking system calls is done via API calls. Because of this structure, it is possible to intercept or monitor system calls (e.g., with a technique sometimes called "hooking"). This can be employed for purposes ranging from malicious (e.g., viruses, malware) to benign (e.g., performance monitoring, anti-malware).

> API spying utilities are among the most powerful tools for exploring the inner structure of applications and operating systems. Unfortunately, neither the SDK nor the DDK provide any documentation or examples demonstrating a way for implementing such a utility. ***This article will attempt to shed some light on this subject by presenting several techniques for hooking API calls issued by Windows applications.***
>
> There are several things you'll need to consider before sitting down to write your own API spying utility. First, you'll need to decide whether you want to spy upon a single application or set up a system-wide API interceptor. Each approach can be useful in different situations. For example, assume that you need to write an application that blocks the execution of specific processes according to rules set by an administrator. Obviously, you will need a way to monitor the execution of new processes and terminate the ones, which are marked as restricted. ***One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually, to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new process, your interceptor will gain control and perform whatever processing is necessary***.
> (*Id.*, p. 1, emphasis added).

69. This can be employed for purposes ranging from malicious (e.g., viruses, malware) to benign (e.g., performance profiling, policy enforcement, activity monitoring).

> There are several things you'll need to consider before sitting down to write your own API spying utility. First, you'll need to decide whether you want to spy upon a single application or set up a system-wide API interceptor. Each approach can be useful in different situations. ***For example, assume that you need to write an application that blocks the execution of specific processes according to rules set by an administrator.*** Obviously, you will need a way to monitor the execution of new processes and terminate the ones, which are marked as restricted. One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually, to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new process, your interceptor will gain control and perform whatever processing is necessary.
>
> Other types of applications might require a simpler API interceptor capable of monitoring just one application at a time. A good example is BoundsChecker from NuMega - a tool capable ***of analyzing API calls in order to detect memory leaks and other bugs lurking inside Windows applications.***
> (*Id.*, p. 1, emphasis added).

70. One particular technique, "interrupt hooking," allows drivers (e.g., device drivers) to be notified when an interrupt occurs that represents a user-mode process making calls to kernel-mode operations:

> In the old days of DOS, interrupt hooking was widely used by TSR (Terminate and Stay Resident) utilities and other applications to extend the operating system functionality and monitor its behavior. Under Windows, interrupts still play a major role, and are mainly used as a portal that connects user-mode (a.k.a. ring 3) code to the operating system's kernel.
> …
> If you have ever examined ntdll.dll with QuickView, you might have noticed that it exports a set of functions that begin with the Nt prefix. These functions are actually small stubs of code that pass control to the Windows NT kernel (NTOSKRNL) using interrupt 2E. Many of the functions exported from kernel32.dll are nothing more than control transfer routines to the stubs located in ntdll. For example, when a Windows application issues a call to CreateFile located in kernel32.dll, the call is redirected to NtCreateFile, which passes it on to NT's kernel for further processing. The special design of this mechanism allows a device driver to hook these interfaces, thus providing a way for ***monitoring activities performed by Windows NT/2000 applications***.
> (*Id.*, p. 6, emphasis added).

C. Processes and files in Windows monitoring

71. The Windows software development kit includes a data structure called Win32_Process, which is part of the Windows Management Instrumentation:

> The Win32_Process WMI class represents a sequence of events on a Windows® operating system. A descendent or member of this class is a sequence that consists of an interaction of one or more processors or interpreters, some executable code, and a set of inputs, for example, a client application running on a Windows system.
> (Ex. 1011, p. 1).

72. Some data (called "properties") available in this structure include "Name", "ProcessId", and "CommandLine". (*Id.*, p. 1). In the Microsoft documentation, "Name" is described simply as "Label for an object." (*Id.*, p. 4). "ProcessId" is described thus: "Global process identifier that you can use to identify a process. The value is valid from the time a process

is created until it is terminated." (*Id.*, p. 6). "CommandLine" is described as "Command line used to start a specific process, if applicable." (*Id.*, p. 2).

73. A POSITA would have understood a "command line" in this context to mean the file name of the executable program and any arguments used when invoking the program. For example, the command line "notepad.exe my_document.txt" includes the file name of the executable ("notepad.exe") and one argument ("my_document.txt"). This command line would start the Notepad program and cause it to open the file "my_document.txt" for editing.

74. Given a process identifier, it is possible to retrieve the command line for the process and thus learn the name of the executable file that is running in that process.

> Processes are represented in WMI by instances of the Win32_Process class. This is derived from CIM_Process, which in turn is derived from CIM_LogicalElement, which we have met several times already. The primary key of a Win32_Process object is its Handle property; this value represents a numeric identifier that the operating system assigns to a process when it is created. Every process running on a computer at any given time has a different handle; the operating system uses this number to uniquely identify each process. Hence, you can easily retrieve a WMI object representing a process whose ID you know using code such as this:
>
> ```
> Set refExistingProcess =
> GetObject("winmgmts:Win32_Process.Handle='1000'")
> ```
> (Ex. 1012, p. 3).

75. The data returned by this code will include the properties listed above (e.g., "Name" and "CommandLine"), which means that the file name of an executable can be retrieved based on the identification of the process (e.g., PID or handle). Accordingly, a POSITA would have understood that a file name or path is retrievable from a process ID (PID) such that a PID can be converted into a file name or location. <u>A POSITA who is interested in preserving the file name or location for later analysis (e.g., in conjunction with a PID or system call) would have been</u>

motivated to retrieve and store the information contemporaneously. That is, the data returned by this code can only be obtained while the process in question is active.

D. Behavior monitoring techniques for anti-malware

76. Many techniques for detecting and responding to malware existed at the time of the alleged invention. These included signature scanning (looking for particular sequences in the content of a suspicious file) and integrity checking (using cryptography to verify that a file is intact and correct).

77. Signature-based virus scanning, for example, is described as follows in the 2003 textbook, "Malware: Fighting Malicious Code":

> One of the simplest and most popular ways in which antivirus software detects malicious code is through the use of virus signatures. The antivirus vendors collect malware specimens and "fingerprint" them. Thousands of signatures are gathered together in a database for use in an antivirus scanner. The database of such signatures is distributed to sys terns that require protection. When scanning files for malicious code, antivirus software compares the current file to its signature set and determines whether the file matches a signature of a known malware specimen.
> (Ex. 1009, pp. 8-9).

78. However, this technique has limitations, namely around the timely creation and propagation of signature databases:

> One of the biggest challenges to this signature-based method of malware detection is that antivirus software needs to include a signature for the virus to discover it on the victim's system. This means that antivirus vendors strive to collect new virus samples, develop patterns that fingerprint them, and distribute signature updates to the customers as quickly as possible.
> (Ex. 1009, p. 9).
>
> Still, even with rapid and frequent updates, one of the weaknesses of the signature-matching approach is that it is always playing catch-up with malware authors as they release brand new, or even slightly modified, specimens into the wild. Additionally, a bad guy might create a custom virus, keeping it close to the vest before releasing it against a particular target. Without a widespread release, antivirus software developers cannot create a signature until the virus has been deployed against its target, which might be too late to stop major damage. Another significant disadvantage

> of this signature-based detection technique is that, in its pure form, it cannot identify malicious code designed to automatically change itself as it propagates, thereby modifying itself so that it doesn't match any signatures. (Ex. 1009, pp. 10-11).

79. Integrity scanning works in the following manner:

> When defending against viruses, we are dealing with creatures that modify their host programs as they spread. Therefore, one way to detect the presence of a virus is to discover files that have been unexpectedly modified. The integrity verification process aims to achieve this goal by following these steps:
>
> 1. While the machine is in a pristine state, compute fingerprints (in the form of checksums or cryptographic hashes) of files that need to be monitored, and record them in a baseline database.
> 2. When scanning the file system for suspicious modifications, compute fingerprints of monitored files and compare the values to those in the baseline.
> 3. If unexplained differences between the current state and the baseline are detected, issue an alert.
> (Ex. 1009, p. 13).

80. Another technique is behavior monitoring. For example, Skoudis describes behavior monitoring using emulation:

> Realizing the limitations of signature-based detection methods, antivirus vendors have devised similar ways in which ***they can detect previously unseen viruses that exhibit certain behavioral and structural characteristics.*** Symantec, for instance, calls this feature of its Norton AntiVirus product Bloodhound. A heuristics-based detection engine scans the file for features frequently seen in viruses, such as these:
>
> - Attempts to access the boot sector.
> - Attempts to locate all documents in a current directory.
> - Attempts to write to an EXE file.
> - Attempts to delete hard drive contents.
>
> As the heuristics scanner examines the file, it usually assigns a weight to each virus-like feature it encounters. If the file's total weight exceeds a certain threshold, then the scanner considers it malicious code. If the scanner's developer sets the threshold too low, then the user could be overwhelmed with false alarms. On the other hand, if the threshold is set too high, or if virus-like features are not properly identified, then the

> detector will miss too many viruses. Either way, the user's protection is limited unless the sensitivity is set just right.
>
> This technique would not be very helpful if antivirus software was able to detect malware only after the virus exhibited malicious behavior such as infecting programs or deleting files. If that were the case, you might get a warning from the antivirus software that says, "Your system has just been completely undermined by a virus! Have a nice day." Although this is certainly interesting information, you need to get the warning before the malware has its way with your machine. The trick is to parse the suspicious file in a way that allows antivirus software to estimate what actions would be performed if the virus actually has a chance to execute. This analysis must occur before the code runs. Antivirus software accomplishes this goal by attempting to emulate the processor that would have executed the potentially malicious program. In the case of executables compiled for Intel x86 machines, this approach calls for emulating key features of the x86 processor. In the case of VBScript macros embedded into Microsoft Office documents, this approach requires emulating basic functionality of the VBScript processing engine.
> (Ex. 1009, pp. 11-12, emphasis added).

81. Also described is a technique called "behavior blocking", which allows suspicious code to execute until it attempts to perform actions that identify it as malicious:

> Instead of relying on signatures to detect known malware specimens, ***behavior blockers keep an eye out for suspicious actions to recognize and block behavior typically associated with malicious code.*** For example, attempts to perform the following actions could suggest that the program is malicious:
>
> - Writing to sensitive portions of the registry
> - Creating a file on a system directory
> - Overwriting browser settings
> - Adding a browser plug-in
>
> (Ex. 1009, p. 24, emphasis added).

82. Indeed, tools to do such behavior blocking were commercially available at the time:

> One tool that uses a behavior-based technique to protect workstations against malicious mobile code is Finjan SurfinGuard, which you can purchase from www.finjan.com. SurfinGuard runs in the background on your machine, creating a tight sandbox around programs and browser scripts that would otherwise have unrestrained access to the system's resources. For example, Figure 4.18 shows a warning that SurfinGuard popped up when a malicious site managed to push a malicious ActiveX

control to a system in my lab. The control slipped through Internet Explorer's security settings because I purposefully misconfigured them—this resembles an environment where a careless user agreed to install an untrusted ActiveX control, or where the browser contained an unpatched vulnerability.

My browser downloaded the ActiveX control and attempted to execute a program named Loader.exe. Fortunately, SurfinGuard alerted me about this activity, giving me an option to block the action or to continue monitoring the suspicious executable. Without this layer of protection, my slightly misconfigured workstation would fall victim to the malicious ActiveX control. Similar behavior-blocking functionality exists as part of the Tiny Personal Firewall (TPF) product sold by Tiny Software at www.tinyspftware.com. TPF allows the user to categorize programs into groups based on their trustworthiness, with each group subject to different restrictions. When protecting the misconfigured machine that I described in the previous example, TPF offered to execute the suspicious program within a sandbox appropriate for the Restricted Applications group, as you can see in Figure 4.19.

Behavior-monitoring tools such as SurfinGuard and TPF help protect workstations against malicious mobile code, because they can often recognize a new malware specimen without relying on signatures that might not even exist for many mobile specimens. In addition to supplementing traditional antivirus software, behavior monitors play well with tools that specifically target spyware programs.
(Ex. 1009, pp. 24-26).

83. Screenshots of the cited figures 4.18 and 4.19 are reproduced hereinbelow:

(Ex. 1009, p. 25, Figure 4.18).

(Ex. 1009, p. 26, Figure 4.19).

84. The work of researchers such as Forrest, Hofmeyr, and Somayaji in the 1990's proposed immunological approaches to malware detection and prevention (i.e., designing systems that function similarly to the human immune system, in which native entities and behaviors are tolerated while foreign pathogens are not), in which normal and abnormal behavior models (which they call a "behavioral signature") can be developed over time and allow for detecting "foreign" (i.e., emergent) threats.

> Do the normal databases allow the system to discriminate between normal and abnormal behavior? To date, we have constructed databases of normal behavior for three Unix programs: sendmail, wu.ftpd (a Linux version of ftpd), and lpr. When comparing the normal database for one program (e.g., sendmail) with traces of normal behavior of a different program (e.g., ls), we observed 40%–80% mismatches over the length of the foreign (e.g., ls) trace. We also observed clear detection of several common intrusions for

> the three programs (mismatch rates generally ranged from 1%–20% for the length of the trace). ***These results suggest that short sequences of system calls do provide a compact signature for normal behavior and that the signature has a high probability of being perturbed during intrusions***.
>
> Although this method does not provide a cryptographically strong or completely reliable discriminator between normal and abnormal behavior, it is much simpler than other proposed methods and could potentially provide a lightweight, real-time tool for continuously checking executing code.
> (Ex. 1014, p. 6, emphasis added).

85. Thus, it was well understood in the art that signature scanning and integrity checking, although useful tools, were not the only anti-malware techniques and that they had limitations (e.g., inability to detect novel malware for which no signature is available). It was also well understood in the art that certain behaviors tend to indicate the presence of malware, and that such behaviors can be monitored or even blocked in order to detect and/or prevent malware.

VI. OVERVIEW OF THE CHALLENGED PATENT

A. THE '243 PATENT SPECIFICATION AND CLAIMS

86. U.S. Patent No. 8,201,243 to Matthew L. Boney ("the '243 patent") is titled "Backwards Researching Activity Indicative of Pestware" and is directed to identifying the source of pestware activity on a computer:

> A system and method for researching an identity of a source of activity that is indicative of pestware is described. In one embodiment the method comprises monitoring the computer for activity that is indicative of pestware, identifying, based upon the activity, an object residing on the computer that is a suspected pestware object; and accessing at least a portion of a recorded history of sources that the computer received files from so as to identify a reference to an identity of a particular source that the suspected pestware object originated from.
> (Ex. 1001, Abstract).

87. To do so, the alleged invention monitors the computer for suspicious activity indicative of pestware, such as changes to registry settings or spawning of new processes. This monitoring can be done by kernel-mode drivers that log API calls.

> Pestware and pestware activity can also be identified by the shields 120, which generally run in the background on the computer system. In the exemplary embodiment depicted in FIG. 1, the shields 120 are divided into the operating system shields 120A and the browser shields 120B. The Shields 120 are designed to watch for pestware and for typical pestware activity and includes two types of shields: behavior-monitoring shields and definition-based shields.
>
> As an example, the shields 120 monitor the protected computer 100 for certain types of activities that generally correspond to pestware behavior. Particular examples of some of the types of activities that are monitored include a process spawning another process, an alteration to registry entries, communications with remote sites via the Internet, alterations to a start up folder, injection of a DLL into another process, and a change to the browser's home page and/or bookmarks. Some specific examples of shields are disclosed in the above identified application entitled System and Method for Locating Malware and Generating Malware Definitions. In the exemplary embodiment, the shields 120 inform the heuristics engine 108 about the activities and the heuristics engine 108 determines whether the research module 108 should be informed and/or whether the activity should be blocked.
> (Ex. 1001, 4:22-44).

88. The data used for detecting pestware may be stored in various types of files:

> In addition, the file storage device 218 includes a browser history 240, browser cache 242, browser settings 244, OS settings 246, an event log 248, a debugging log 250, a firewall log 252, file information 254 and monitoring software logs 256. One or more of these exemplary files 240-256 may be implemented for one or more of the N files 130 described with reference to FIG. 1.
> (Ex. 1001, 6:17-23).

89. The heuristics engine analyzes the monitored activity, assigning weighted factors to assess whether the activity exceeds a threshold to be considered pestware:

> As an example, the heuristics module 110 may compare the activities on the computer with weighted factors to make decisions relative to activities at the protected computer. Each of these factors may be, for example, associated with a particular activity and each factor may be weighted by the likelihood that the activity is associated with pestware. If the sum of the weighted factors that match the activity history exceed a threshold, then the activity is identified as pestware activity and the heuristics module 110 prompts the research module 108 to initiate research into the origin of the pestware initiating the activity.

(Ex. 1001, 4:45-55).

90. Thereafter, the research module uses logged historical data to identify the external origin of the pestware (e.g., a URL):

> In accordance with several embodiments, the research module 108 is configured to receive an indication that either known pestware is residing on the computer or activities indicative of pestware have occurred or are occurring on the protected computer. In response, the research module 108 is configured to research the activity log 120 and/or the N files 130, which include information (e.g., historical logs) relating to events on the computer 100, to identify a source of the pestware 122, 124 or pestware-related activities, which may be reported by the reporting module 114 to a centralized data store for subsequent pestware management purposes.
> (Ex. 1001, 3:30-40).

91. The reporting module can then report this identity to an external research entity:

> For example, the identities (e.g., I.P. address, URL, email client or program name) of sources (e.g., web sites, email or program) of the pestware may be collected in a centralized data store (not shown) and then subsequently reported to other users. In addition, the sources of the pestware or suspected pestware may be visited to further research how the pestware is being distributed from the URLs and/or to generate new definitions for pestware discovered at these sources.
> (Ex. 1001, 3:41-48).

B. PROSECUTION HISTORY

92. In the non-final Office Action that issued on February 25, 2010, all pending claims (1-22) were rejected. Claims 1-22 were rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent Publication 2005/0268112 to Wang et al. ("Wang") in view of U.S. Patent Publication 2004/0024864 to Porras et al. ("Porras"). (Ex. 1002, p. 178). Applicants responded to the non-final Office Action by amending the independent claims. Specifically, the independent claims were amended to recite "externally networked" and "particular externally networked". (*Id.*, pp. 160–65). In associated remarks, Applicants argued that Porras is not interested in "an externally networked source suspected of originating an attack...but in the internal network's user or process

session through which the attack is made". (*Id*. p. 167). Applicants stated that the other cited references fail to cure the deficiencies of Porras. (*Id*.).

93. In the final Office Action, issued November 10, 2010, all pending claims were rejected. (*Id*., p. 136). Claims 1-22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Wang in view of U.S. Patent Application No. 2007/0168694 to Maddaloni et al. ("Maddaloni"). Applicants responded to the final Office Action with a Request for Continued Examination by not amending the claims and arguing that Maddaloni is ineligible prior art pursuant to 35 U.S.C. § 103(c)(1). (*Id*., p. 112).

94. In the non-final Office Action that issued on May 9, 2011, all pending claims (1-22) were rejected. Claims 1-22 were rejected under 35 U.S.C. § 103(a) as unpatentable over Wang in view of U.S. Patent No. 7,089,428 to Farley et al. ("Farley"). (*Id*., p. 85). Applicants responded to the non-final Office Action by amending claim 1 to recite, in part "reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware", and claim 17 to recite, in part "reporting the identity of the particular web site to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware". (*Id*., pp. 64-68).

95. In the Notice of Allowability, issued February 9, 2012, the Examiner provided an Examiner's Amendment previously agreed to by the Applicant in an interview amending claim 1 to recite, in part "monitoring, <u>using a kernel-mode driver</u>...<u>wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log; analyzing, heuristically, computer activity to determine whether one or more weighted factors</u>

associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware...accessing, in response to the identifying an object.", and amending claim 17 (now claim 12) to recite, in part "monitoring, with a kernel-mode driver...wherein the instructions for monitoring include instructions for monitoring API calls and storing a history of at least a portion of the API calls in an activity log; analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware...accessing, in response to the identifying an object". (*Id*., pp. 47-49). The patent subsequently issued.

VII. LEVEL OF SKILL IN THE ART

96. I understand that Petitioners submit that a person of ordinary skill in the art ("POSITA") at the time of the earliest claimed priority date for the '243 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 topics such as malware/exploit detection and/or machine learning. More education can supplement practical experience and vice versa.

97. I am at least a person of ordinary skill in the art and I have worked with persons of ordinary skill in the art as part of my professional responsibilities.

VIII. OVERVIEW OF THE PRIOR ART

A. OVERVIEW OF ABAD

98. "Log Correlation for Intrusion Detection: A Proof of Concept" is a 2003 research paper by Cristina Abad et al. ("Abad"). It presents a log correlation method to improve intrusion detection systems. In particular, the paper demonstrates that correlating multiple types of logs together improves detection accuracy:

> We argue the need for correlating data among different logs to improve intrusion detection systems accuracy. We show how different attacks are reflected in different logs and argue that some attacks are not evident when a single log is analyzed. We present experimental results using anomaly detection for the virus Yaha. Through the use of data mining tools (RIPPER) and correlation among logs we improve the effectiveness of an intrusion detection system while reducing false positives.
> (Ex. 1006, p. 1).

99. Abad teaches the use of a data mining tool called RIPPER, which ingests labeled training data and infers rules based on that data. In Abad's system, some of the data used to train RIPPER consists of series of system calls:

> RIPPER expects a list of system calls followed by the value that it will predict. For example, if we decide that we are going to use five system calls to predict the sixth, then RIPPER expects the five system calls followed by the sixth that it will try to predict. A sliding window was used to scan the normal traces and create a list of unique sequences of system calls.
> (Ex. 1006, p. 6).

> After processing the training data, RIPPER generates a series of rules. One such rule may be: if system call 1 is Open, then system call 7 should be Closed (with a confidence level of 0.38).
> (Ex. 1006, p. 5).

100. These rules can be validated against testing data to calculate a confidence level for each rule (i.e., how likely was the rule to be correct on the testing data):

> After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples.
> (Ex. 1006, p. 5).

> Approximately 20% of the captured data was saved to use for testing only; i.e., it was not used in training. The generated rules were applied to the testing data to determine how abnormal the traces were. RIPPER returned a list that consists of each predicted result (predicted based upon the rules) along with the actual value. It also gives the confidence level for the particular rule that was used in the prediction.
> (Ex. 1006, p. 6).

101. The rules can then be applied to new data (e.g., collected from activity on a live system), weighted by the confidence levels, and thus determine a classification of the activity:

> A sliding window of size 13 was passed over the list of predicted values and actual values returned by RIPPER. If the predicted value differs from the actual value, the penalty value is set to the confidence level of the rule that was broken. Then the penalty values are summed from the 13 spaces of the sliding window and if the value is greater than half (6.5), this "region" is considered as abnormal, otherwise it is considered as normal.
> (Ex. 1006, p. 6).

102. Abad teaches the use of two types of logs: system calls and network activity:

> Looking at the system call sequence, abnormal behavior may be observed which is not deemed statistically significant to signify an intrusion in a tuned IDS. However, if we examine the corresponding network traffic the traces show an unusual amount of network traffic corresponding to the system call sequence where the user opens the e-mail. Combining this information from the system call sequence and the network traffic, it is possible to identify intrusions that would normally not be identified using just one log.
> (Ex. 1006, p. 5).

103. Abad also demonstrates training RIPPER on network traffic data:

> Taking an idea from Lee and Stolfo's work [15], in order to get meaningful results for network traffic it is necessary to introduce some type of temporal information into the equation. Therefore, we decided to introduce the number of connections within the past 10 seconds as a data item. The rationale behind this decision is that a sudden spike in network traffic could be a good sign of an intrusion. In short, RIPPER used the characteristics from the network traffic connections in order to predict how many network connections have occurred in the past 10 seconds.
> (Ex. 1006, p. 6).

104. The goal of the experiments was to determine whether correlating multiple types of logs would lead to improved detection accuracy:

> The main goal of our experiments was to test if this method may improve intrusion detection systems when the results from multiple logs are correlated (specifically traffic logs and system call logs). For example, if there is a slight anomaly in network traffic the confidence levels may change when mining the system calls. Unusual behavior found in a network traffic log may be an indication of an intrusion so the confidence level of

> certain rules for the system call log may be increased. Thus, we are able to identify intrusions with higher confidence levels that would normally not be identified by using a single log (traffic or system log).
> (Ex. 1006, p. 6).

105. Abad notes that the inputs to the system often have high variability and thus have increased chances of false positive results:

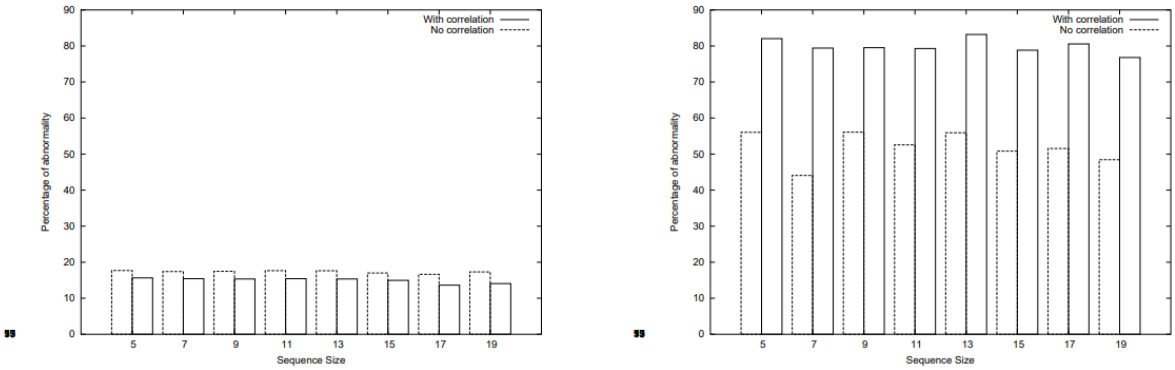
> Although our results show an improvement in anomaly detection, we also determined that the "predict the next system call" method does not perform well for this particular problem. Normal traces had a high level of abnormal behavior to begin with, increasing the chance of false positives.
> (Ex. 1006, p. 9).

106. Nevertheless, Abad determined that this kind of correlation did improve intrusion detection accuracy:

> Our experiments produced interesting results. As is observable from the tables, the results show that correlation increases the accuracy of the intrusion detection system. After correlation, normal traces become more normal while abnormal traces become more abnormal (See Figure 2). We believe that our data mining techniques could be applied to other IDSs to increase the accuracy of intrusion detection.
> (Ex. 1006, pp. 7-9).

> Specifically in this paper, we performed experiments to successfully correlate data from multiple logs in a laboratory setting for a particular case of anomaly detection (the Yaha virus). We report empirical results showing improved IDS accuracy by correlating network log and system call information. Not only were we able to make abnormal traces more abnormal, thus making intrusions easier to detect, but we were able to decrease false positives by making normal traces more normal.
> (Ex. 1006, p. 9).

107. Figure 2 shows traces without correlation on the left and with correlation on the right, for both normal traces and abnormal traces:



(a) A normal behavior trace (b) An abnormal behavior trace

(Ex. 1006, p. 9, Figure 2).

B. OVERVIEW OF PICCARD

108. U.S. Patent Publication No. 2007/0006310 to Paul L. Piccard, "Systems and Methods for Identifying Malware Distribution Sites" ("Piccard"), is directed toward identifying the source of malware that has been detected. It does so through the use of several modules. The "malware detection module" is responsible for identifying a suspicious file. Piccard describes several exemplary ways this can be done, including by monitoring for suspicious activity patterns:

> The first operation illustrated in FIG. 2 is to analyze a file to determine that the file includes potential malware (block 200). In the illustrated embodiment, a malware detection module (e.g., the malware detection module 118) scans files of a protected computer (e.g., the protected computer 102) to locate the file. The malware detection module can scan files of the protected computer on a periodic or some other basis. Alternatively, or in conjunction, operation of the malware detection module can be triggered based on determining that the file is being downloaded or has been downloaded using a Web browser (e.g., the Web browser 114).
>
> In the illustrated embodiment, the malware detection module compares the file with a set of malware definitions to determine if the file includes potential malware. The set of malware definitions can include representations of malware, suspicious activities that are indicative of or that are common to malware, or both. For example, the set of malware definitions can include a hash value or a digital signature of malware. Such as one that is generated using Message Digest 5 ("MD5"). In this example, the malware detection module generates a hash value for the file and

> compares the hash value of the file with a set of hash values of malware to determine whether there is a sufficient match. As another example, the set of malware definitions can include a Cyclical Redundancy Code ("CRC") of a portion of malware. In this example, the malware detection module generates a CRC for the file and compares the CRC of the file with a set of CRCs of malware to determine whether there is a sufficient match. ***As a further example, the set of malware definitions can include suspicious activities related to third-party cookies or related to entries or modifications of registry files of an operating system.***
> (Ex. 1007, [0019]-[0020], emphasis added).

109. Once a suspicious file has been identified, the "Web site identification module" is responsible for determining the source of the file. The browser history log contains information about the source of downloaded files:

> As illustrated in FIG. 1, the memory 112 also stores a history log 116, which is maintained by the Web browser 114 to provide a record of browsing events. In particular, when a file is accessed and downloaded, the Web browser 114 records a Web address of the file in the history log 116. A Web address typically specifies a location of a file within a Web site. For example, a Web address can be a Uniform Resource Identifier ("URI") of a file, such as a Uniform Resource Locator ("URL") of the file. It is also contemplated that a Web address can be defined in various other ways, such as using an Internet Protocol ("IP") address or any other identifier of a source of a file.
> (Ex. 1007, [0013]).

110. Using this information, the "Web site identification module" can identify the source of the suspicious file:

> ***The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded*** (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file, which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where

> "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. ***Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.***
> (Ex. 1007, [0021], emphasis added).

111. Thereafter, Piccard teaches a "reporting module" responsible for notifying a "remotely-located computer" of the website, which may be a malware distribution site, in order to prompt further research about the website:

> The third operation illustrated in FIG. 2 is to report that the Web site is a potential malware distribution site (block 204). In the illustrated embodiment, once the Web site identification module identifies the Web site, ***a reporting module (e.g., the reporting module 122) reports information regarding the Web site to a remotely-located computer that is connected to the protected computer.*** This information can identify the Web site as a potential malware distribution site, such as in terms of the domain name of the Web site. Alternatively, or in conjunction, this information can identify the file as including potential malware, such as in terms of the name of the file or the URL of the file. It is also contemplated that this information can include a representation of the file or can identify suspicious activities related to the file. ***This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the file does, in fact, include malware and whether the Web site is, in fact, a malware distribution site.*** If the Web site is determined to be a malware distribution site, a new or updated set of malware definitions can be generated based on content within the Web site, and the new or updated set of malware definitions can be provided to the protected computer. In addition, content within the Web site can be monitored on a periodic or other basis for new or updated malware. Also, a new or updated list of malware distribution sites can be generated so as to identify the Web site, and the new or updated list of malware distribution sites can be provided to the protected computer.
> (Ex. 1007, [0023], emphasis added).

C. OVERVIEW OF KILPATRICK

112. U.S. Patent No. 6,735,703 to Kilpatrick et al. ("Kilpatrick") is titled "Multi-Platform Sequence-Based Anomaly Detection Wrapper" and is directed to real-time, efficient anomaly detection through monitoring system calls made by an executable:

> The present invention meets the aforementioned needs by providing an intrusion detection system that operates efficiently in real-time. Computational efficiency is generated through the representation of known sequences of system calls in a distance matrix. The distance matrix indirectly specifies known sequences by specifying allowable separation distances between pairs of system calls. The distance matrix is used to determine whether a sequence of system calls in an event window represents an anomaly. Anomalies that are detected are further analyzed through levenshtein distance calculations that also rely on the contents of the distance matrix. In a preferred embodiment, the intrusion detection system is incorporated as part of a system call software wrapper. It is a further feature of the present invention that event abstraction enables the intrusion detection system to apply generically across various computing platforms.
> (Ex. 1008, 3:7-23).

113. Kilpatrick analyzes sequences of system calls, which are intercepted through the use of a "wrapper":

> In accordance with the present invention, a software wrapper provides a dynamically installable, non-bypassable, operating system extension that intercepts a selected program's system calls. For each system call it intercepts, a wrapper may perform functions such as s (1) add functionality (intrusion detection, auditing, encryption) to the system call, (2) transform the system call (e.g. by replacing it with another), or (3) prevent the system call's execution (e.g., perform access control).
> (Ex. 1008, 4:17-25).

> As illustrated in FIG. 2, operating within the UNIX environment are programs 202A and 202B. Program 202B is an unwrapped program whose system calls pass directly to an internal application programming interface (API). Program 202A, on the other hand, is "wrapped" by software wrapper 210. As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210 based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).

> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. ***These wrapper instances wrap their processes by intercepting some or all of the system calls that are made by the process.*** This interception effectively puts the wrappers in complete control of their processes' interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:33-55, emphasis added).

114. The system calls thus intercepted are stored in an "event window" of a certain width:

> System call event window 300 represents a sequence of system call events that are moving through system call event window 300 from left to right. System call event window 300 is fixed in size and can be arbitrarily set to a width of N events. In a preferred embodiment, system call event window 300 contains eight events at locations designated as positions A-H.
> (Ex. 1008, 5:66-6:5).

115. This event window is used during two modes: "record mode" and "detect mode". During the former, a distance matrix is created that represents the various system call pairs that were observed during routine operation of the program, as well as how far apart such pairs were seen.

> In the present invention, a distance matrix (illustrated as reference number 220 in FIG. 2) is used to represent the various observed event sequences. Distance matrix 220 includes information on pairs of events by storing the known distances that a first system call follows a second system call. For a program that includes M possible system calls, and an event window 300 having a width of N events, distance matrix 220 can be represented by a MxMxN data array structure. For example, in a 256x256x8 data structure, the value of data element (25, 37, 1) would signal whether system call 37 has been known to follow system call 25 at a distance of one (i.e., immediately following).
>
> The content of the MxMxN data array structure can be used to represent all possible system call event pairs that can appear in event window 300. It should be noted that the collection of system call event pairs indirectly define the various known system call sequences.
> (Ex. 1008, 6:29-45).

116. During "detect mode", the event window is populated with system calls made by the monitored executable, and the pairs in the event window are compared with the contents of the distance matrix.

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
>
> At this point, the system determines whether the new sequence of system calls in event window 300 is a recognized sequence. In accordance with the present invention, the determination process is not based upon a literal comparison of entire sequences of system calls. Rather, multiple pairs of system calls are compared relative to the predefined contents of distance matrix 220.
>
> More specifically, at step 506, a first pair of system calls is selected. As illustrated in FIG. 3, the first pair of system calls is represented by new system call X at position A and the previously received system call 51 at position B. At step 508, the software wrapper then determines whether this system call pair relationship is represented in distance matrix 220. More specifically, the system determines whether distance matrix element (51, X, 1) is set to true. If distance matrix element (51, X, 1) is determined, at step 408, to be set to true, then the system call pair relationship has been previously seen. The process then proceeds directly to step 512 where it is determined whether more system call pairs need to be processed.
> (Ex. 1008, 8:33-58).

117. If the system call pair has previously been seen at that distance (e.g., during normal program behavior observed during "record mode"), then no further action is necessary. However, if the system call pair has not been seen at that distance, then additional computation is necessary to determine whether the behavior is abnormal.

> If, at step 508, one of the system call pairs has been determined not to exist in distance matrix 220, then the system call sequence in event window 300 has not been seen before. In this case, the detect process proceeds to step 510, where anomaly analysis processing is performed***. In the anomaly analysis processing step, the magnitude of the anomaly is identified. This magnitude measure represents the extent to which the system call***

> ***sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220.*** In general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion.
>
> In a preferred embodiment of the present invention, a levenshtein distance is used as a measure of the magnitude of the difference between a system call sequence in a stored profile and the system call sequence that is generated by a program.
> (Ex. 1008, 9:8-23, emphasis added).

118. Kilpatrick then discloses that once the "magnitude" or "distance" calculations are complete, the system can determine that the process is abnormal if the distance exceeds a threshold:

> The minimum value across all possible values of test head therefore represents the minimum levenshtein distance across all known sequences of system calls. Based upon the value of the minimum levenstein distance, the intrusion detection system can then determine whether an intrusion alert, or other forms of intrusion detection processing, should be initiated. For example, if the minimum levenstein distance is greater than a predefined threshold, the intrusion detection system can initiate steps to terminate the abnormal process.
> (Ex. 1008, 11:19-28).

IX. GROUND 1: CLAIMS 1-14 ARE RENDERED OBVIOUS BY ABAD IN VIEW OF PICCARD

119. As I describe in greater detail hereinbelow, a POSITA would have found Claims 1-14 rendered obvious by Abad in view of Piccard.

A. Motivation to Combine Abad and Piccard

120. A POSITA would have been motivated to combine the teachings of Abad and Piccard. As I discussed in Section VIII.A., Abad is directed toward detecting malicious or unwanted behavior through the correlation of multiple logs:

> We argue the need for correlating data among different logs to improve intrusion detection systems accuracy. We show how different attacks are reflected in different logs and argue that some attacks are not evident when a single log is analyzed.
> (Ex. 1006, p. 1).

121. Having additional sources of activity information increases prediction accuracy:

> Looking at the system call sequence, abnormal behavior may be observed which is not deemed statistically significant to signify an intrusion in a tuned IDS. However, if we examine the corresponding network traffic the traces show an unusual amount of network traffic corresponding to the system call sequence where the user opens the e-mail. ***Combining this information from the system call sequence and the network traffic, it is possible to identify intrusions that would normally not be identified using just one log.***
> (Ex. 1006, p. 5).

122. In particular, Abad argues the need for activity information from multiple sources, beyond just system calls and network activity, such as server logs, application logs, and firewall logs:

> Another shortcoming of current approaches in intrusion detection systems is that they only attempt to detect and prevent individual attacks and not coordinated distributed attacks. Much of the work centers around improving the ability of systems to detect attacks and the speed of the traffic that can be handled. Both signature-based and anomaly detection sensors are useful tools but they only provide a snapshot of the event in time after it has transpired. On the other hand ***clues as to what is happening in real-time on a computer network is available but spread out all over the network.*** For instance, all modern operating systems can log system and security events. Servers keep extensive records of their operations. Many applications write errors, warnings, and failures to their own logs. Firewalls can track how many packets are dropped by different rules typically at the entry and exit points of the network. ***If all this information can be brought together it would enable more robust attack detection mechanisms.***
> (Ex. 1006, pp. 2-3, emphasis added).

123. Abad thus clearly describes the need for network-wide threat research, while focusing its disclosures on correlation within a single machine as a component of that larger effort:

> We believe that correlating multiple logs in a single host is also important to increase the efficiency of detecting intrusions. We define employing log correlation in a single host as host-based correlation. By integrating different information from multiple logs, it is possible to reduce false alarms and increase detection rate.
> (Ex. 1006, p. 3).

124. Abad also teaches a three-stage approach to computer network security:

> The challenges in securing a computer network can be viewed in three stages [27]:
>
> - Prevention: Avoid intrusions if possible. Vulnerabilities should be identified and patched. It is important to use security measures such as user authentication, authorization and access control.
>
> - Detection: Know as soon as possible when an intrusion occurs. Installing and using well configured anti-virus software, firewalls and IDSs helps early detection.
>
> - Reaction: Respond to an intrusion. Problems should be fixed, intruders stopped and data restored.
>
> (Ex. 1006, p. 2).

125. While Abad's system fits squarely in the "Detection" stage (it is directed toward improving IDSs), it clearly anticipates the need for post-detection response, such as coordinated research or notification to other parties on the network.

126. Piccard, which is in the same field of endeavor, is directed to the identification of malware and its sources:

> Techniques are currently available to detect and remove malware. But as malware evolves, techniques for detecting and removing malware should also evolve. Accordingly, current techniques for detecting and removing malware are not always satisfactory and will likely not be satisfactory in the future. Current techniques for detecting and removing malware often use definitions of known malware to scan files of a protected computer. However, it is often difficult to initially locate malware in order to generate the definitions, particularly since malware can evolve. ***In particular, it would be desirable to identify sources of malware, such that definitions can be generated or updated to account for evolving malware.*** In addition, identification of sources of malware would allow a blacklist of Web sites to be generated.
>
> (Ex. 1007, [003], emphasis added).

127. Piccard teaches a "malware detection module" as the first operation in its method, "to analyze a file to determine that the file includes potential malware (block 200)." (Ex. 1007, [0019]). Several possible techniques for such detection are disclosed, including monitoring for suspicious activity patterns:

> In the illustrated embodiment, the malware detection module compares the file with a set of malware definitions to determine if the file includes potential malware. The set of malware definitions can include representations of malware, suspicious activities that are indicative of or that are common to malware, or both. For example, the set of malware definitions can include a hash value or a digital signature of malware. Such as one that is generated using Message Digest 5 ("MD5"). In this example, the malware detection module generates a hash value for the file and compares the hash value of the file with a set of hash values of malware to determine whether there is a sufficient match. As another example, the set of malware definitions can include a Cyclical Redundancy Code ("CRC") of a portion of malware. In this example, the malware detection module generates a CRC for the file and compares the CRC of the file with a set of CRCs of malware to determine whether there is a sufficient match. ***As a further example, the set of malware definitions can include suspicious activities related to third-party cookies or related to entries or modifications of registry files of an operating system.***
> (Ex. 1007, [0020], emphasis added).

128. A POSITA would have recognized that Abad's teachings of detecting suspected malware through behavior monitoring align with Piccard's disclosures of its "malware detection module". Because Piccard discloses the "malware detection module" as a discrete component with multiple possible implementations but a defined output (i.e., the suspected file), a POSITA would have expected a high degree of success implementing Abad's techniques as part of Piccard's malware detection module.

129. In a combined system, the log aggregator of Abad would aggregate the system call logs and the network logs of its monitors and aggregate the file download logs and browser history logs of Piccard using methods known to a POSITA and described in Abad. (*See* Ex. 1006, p. 5, emphasis added: "We believe that correlating ***multiple logs in a single host*** is also important…By ***integrating different information from multiple logs***, it is possible to reduce false alarms and increase detection rate."). After aggregation and correlation of the download logs and browser history logs with the logs of Abad, a POSITA would have understood that the network access logs

of Abad would be enhanced to relate access history with web site history and the system call logs of Abad would be enhanced to relate file downloads with new or anomalous system calls.

130. Furthermore, because Abad anticipates the value of correlation with multiple types of activity information (*see* Ex. 1006, pp. 2-3), a POSITA would have been motivated to include, as a further source of activity information, Piccard's disclosures of using the web history log to determine the source of the suspected malware.

131. Abad also provides motivation for post-detection reaction to intrusions, for which a POSITA may have sought to use Piccard's "Web site identification module" and "reporting module" to better respond to suspected malware in order to prevent its recurrence. (*See* Ex. 1006, p. 2, emphasis added: "Problems should be fixed, ***intruders stopped***, and data restored.").

132. Abad is also clear that relevant information about malicious activity is available across the network, and that such information should be examined together and used to improve the security of the entire network: "On the other hand clues as to what is happening in real-time on a computer network is available but spread out all over the network. … If all this information can be brought together it would enable more robust attack detection mechanisms." (Ex. 1006, pp. 2-3).

B. Claim 1 is rendered obvious by Abad in view of Piccard

133. Each of the limitations of Claim 1 of the '243 patent is rendered obvious by Abad in view of Piccard as described below.

1. Limitation 1[a]

134. Limitation 1[a] of the '243 patent reads:

> *A method for identifying an origin of activity on a computer that is indicative of pestware comprising:*

135. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 1 of the '243 patent is considered limiting, Abad in view of Piccard discloses the preamble of Claim 1.

136. Abad describes its goal as follows:

> Intrusion detection is an important part of networked-systems security protection. Although commercial products exist, finding intrusions has proven to be a difficult task with limitations under current techniques. Therefore, improved techniques are needed. We argue the need for correlating data among different logs to improve intrusion detection systems accuracy. We show how different attacks are reflected in different logs and argue that some attacks are not evident when a single log is analyzed. We present experimental results using anomaly detection for the virus Yaha. Through the use of data mining tools (RIPPER) and correlation among logs we improve the effectiveness of an intrusion detection system while reducing false positives.
> (Ex. 1006, p. 1).

137. Piccard teaches a method for identifying the source of malware by examining web traffic history: "Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded." (Ex. 1007, [0005]).

138. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection

module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings at least based on the known log aggregation and correlation techniques disclosed in Abad.

139. Thus, Abad in view of Piccard discloses limitation 1[a].

2. Limitation 1[b]

140. Limitation 1[b] of the '243 patent reads:

> *monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;*

141. Abad discloses limitation 1[b] of the '243 patent.

142. Abad teaches monitoring system calls, such as Windows API calls, at the kernel level: "Our architecture consists of logging utilities that monitor system activity from different vantage points. For example, APISpy32 [10] monitors the system from the kernel level by recording the system calls executed by the system." (Ex. 1006, p. 6). Further, Abad states "[t]he logged system calls and network traffic on the client machine were then mined to determine if an intrusion had taken place." (Ex. 1006, p. 5). As I discussed in Section V.B., a POSITA would have understood that Windows system calls, such as those being recorded by a monitor like APISpy32, relate to the claimed "API calls" of this limitation. (*See* Section V.B.). For example, an API call of CreateProcess indicates one process (perhaps a malicious process) attempting to spawn another process, and monitoring calls to that API would allow analysis of such behavior:

> One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually,

> to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new process, your interceptor will gain control and perform whatever processing is necessary.
> (Ex. 1013, p. 1).

143. Thus, Abad discloses limitation 1[b].

3. Limitation 1[c]

144. Limitation 1[c] of the ’243 patent reads:

> *analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;*

145. Abad discloses limitation 1[c] of the ’243 patent.

146. Abad, during its training phase, creates rules that predict the next expected system call in each sequence of system calls:

> RIPPER expects a list of system calls followed by the value that it will predict. For example, if we decide that we are going to use five system calls to predict the sixth, then RIPPER expects the five system calls followed by the sixth that it will try to predict. A sliding window was used to scan the normal traces and create a list of unique sequences of system calls.
> (Ex. 1006, p. 6).

147. Each rule also has an associated confidence level that indicates how widely the rule applies among the training data:

> After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples.
> (Ex. 1006, p. 5).

148. Using these rules and confidence levels, Abad teaches using monitored activity to determine whether the activity is anomalous (e.g., “abnormal”).

> Approximately 20% of the captured data was saved to use for testing only; i.e., it was not used in training. The generated rules were applied to the

> testing data to determine how abnormal the traces were. RIPPER returned a list that consists of each predicted result (predicted based upon the rules) along with the actual value. It also gives the confidence level for the particular rule that was used in the prediction.
> (Ex. 1006, p. 6).

149. Abad compares the actual observed system call sequences with the rules generated by RIPPER (i.e., the "predicted values") and calculates penalty values which, when summed and compared with a threshold, represent how normal or abnormal a sequence was:

> A sliding window of size 13 was passed over the list of predicted values and actual values returned by RIPPER. If the predicted value differs from the actual value, the penalty value is set to the confidence level of the rule that was broken. Then the penalty values are summed from the 13 spaces of the sliding window and if the value is greater than half (6.5), this "region" is considered as abnormal, otherwise it is considered as normal.
> (Ex. 1006, p. 6).

150. Abad teaches that having weighted factors (e.g., "confidence level" of rules) is desirable, to increase the likelihood of correct classification:

> The reason we chose to use the confidence level as a penalty value is to avoid rules having equal weight. For example, a rule that has a confidence level of 50% should not have the same penalty cost as a rule with confidence level of 99%.
> (Ex. 1006, p. 7).

151. In addition, Abad teaches heuristically applying thresholds when correlating activity from system calls with activity from network traffic, to determine whether the overall activity during a given time period is "abnormal". For example, the threshold value of 6.5 cited earlier may be increased or decreased based on some of the activity:

> Figure 1 illustrates the correlation process. Figure 1(a) shows how the sliding window is used to detect if a region is abnormal. In Figure 1(a) the region is normal, since there are only 3 abnormals and the threshold is 3.5. In Figure 1(b) we lower the threshold by 1. Notice that the same region is now considered abnormal. Lowering the threshold is equivalent to increasing suspicion.

> The results from the previous step were used in processing the system calls again, by following the same steps in Section 8.6 except that the time sequences that were determined were used along with their penalty values in correlating the data. The penalty value was used to adjust the threshold to determine if a region is "abnormal".
>
> Then the system calls were processed. ***When there was a sequence with a time value between 1pm and 1:10pm the threshold was lowered by the penalty value (3). This has the effect of increasing the suspicion level based on the data in a different log. After the correlation, it only took a 3.5 value to declare a region as abnormal as opposed to the original 6.5 value.***
>
> We also decided to increase the threshold during normal time periods. We expect this to help to eliminate false positives. ***For example, if network traffic is normal between 1:10pm and 1:30pm we can increase the threshold to detect an abnormal region by 1. So, it would take 7.5 to label a region as abnormal during that time period.***
> (Ex. 1006, p. 7, emphasis added).

152. Figure 1 illustrates the effect of the threshold on the outcome:

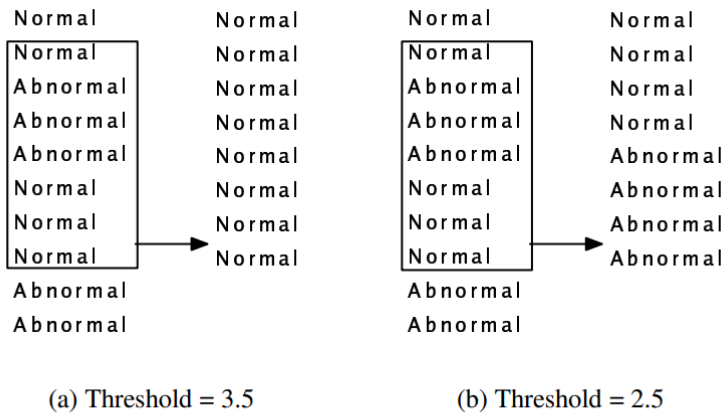


Figure 1. Sliding window in the correlation process

(Ex. 1006, p. 7, Figure 1).

153. A POSITA would have understood Abad's use of penalty values and thresholds for inferring the normal or abnormal status of a region to represent a type of heuristic: making an educated guess based on indirect or complementary evidence.

154. Thus, Abad discloses limitation 1[c].

4. Limitation 1[d]

155. Limitation 1[d] of the ’243 patent reads:

identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;

156. Abad discloses limitation 1[d] of the ’243 patent.

157. Abad discloses monitoring activity logs, such as system call logs:

> Our architecture consists of logging utilities that monitor system activity from different vantage points. For example, APISpy32 [10] monitors the system from the kernel level by recording the system calls executed by the system.
> (Ex. 1006, p. 6).

> We used APISpy32 [10], with some modification, to intercept system calls. The logged system calls and network traffic on the client machine were then mined to determine if an intrusion had taken place.
> (Ex. 1006, p. 5).

158. A POSITA would have understood that system call logs identify the process that made the system call. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity (e.g., the claimed “suspected pestware object”), and a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs.

159. Abad describes using the collected data to predict the next system call in a system call sequence, in order to determine whether there was an intrusion:

> Then second approach [i.e., in which RIPPER is trained with “normal” traces but not “abnormal” traces] uses a system call sequence to predict the next system call. After processing the training data, RIPPER generates a series of rules. One such rule may be: if system call 1 is Open, then system call 7 should be Closed (with a confidence level of 0.38). After RIPPER

> generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples. When mining the actual experimental data from both the network traffic log and system call log, the confidence level of the violated rule is adjusted similar to Lee et al. [14]. The averaged score (by the total number of sequences) of the trace is used to decide whether there is an intrusion.
> (Ex. 1006, pp. 5-6).

160. Thus, Abad discloses limitation 1[d].

5. Limitation 1[e]

161. Limitation 1[e] of the ’243 patent reads:

> *accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and*

162. Abad in view of Piccard discloses limitation 1[e] of the ’243 patent.

163. Abad teaches that identifying a suspected intrusion from a single data source (e.g., system call logs) is less effective alone than correlating that suspected intrusion with data from one or more additional sources (e.g., network traffic logs), and Abad provides motivation for seeking such correlations:

> We believe that correlating multiple logs in a single host is also important to increase the efficiency of detecting intrusions. We define employing log correlation in a single host as host-based correlation. By integrating different information from multiple logs, it is possible to reduce false alarms and increase detection rate.
> (Ex. 1006, p. 3).

164. Abad discloses correlating network traffic logs, which a POSITA would have understood to represent “a recorded history of externally networked sources that the computer

received files from", in addition to being a record of any local traffic the computer may have received:

> Since it is unlikely that we will ever be able to prevent all intrusions, being able to detect an intrusion is fundamental. ***As long as traffic flow is allowed between an internal network and the Internet,*** the opportunity for an attacker or worm to penetrate will always exist. Despite the preventive steps taken, intrusions that involve new techniques usually succeed. ***During and after an attack, network and system logs are the primary sources of information for detection, response, and forensics, without these logs there is little chance of ever discovering what happened.***
> (Ex. 1006, p. 2, emphasis added).
>
> Looking at the system call sequence, abnormal behavior may be observed which is not deemed statistically significant to signify an intrusion in a tuned IDS. However, if we examine the corresponding network traffic the traces show an unusual amount of network traffic corresponding to the system call sequence where the user opens the e-mail. Combining this information from the system call sequence and the network traffic, it is possible to identify intrusions that would normally not be identified using just one log.
> (Ex. 1006, p. 5).

165. A POSITA would have understood that system call logs identify the process that made the system call. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity (e.g., the claimed "suspected pestware object"), and a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs.

166. Abad also describes collecting network logs and network activity / traffic to assist in tracing intrusions. (*See* Ex. 1006, pp. 3, 5). After aggregation and correlation of the download logs and browser history logs of Piccard with the logs of Abad, a POSITA would have understood that the network access logs of Abad would be enhanced to relate access history with web site

history, and the system call logs of Abad would be enhanced to relate file downloads with new or anomalous system calls.

167. Piccard also teaches this limitation in its disclosures about two of the modules in its system. One, a "malware detection module," performs the claimed "identifying an object" (*see* my analysis of limitation 1[d]). Thereafter, Piccard's "Web site identification module" attempts to identify the external source of the malware:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

168. In more detail, Piccard discloses that the "Web site identification module" is invoked after a determination of malware by the "malware detection module", and that it accesses the history log to determine the external source:

> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

169. Piccard's "history log" teaches the claimed "a recorded history of externally networked sources that the computer received files from", and "a Web address from which the file was received" teaches the claimed "a reference to an identity of a particular externally networked source that the suspected pestware object originated from".

170. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings. Furthermore, Abad teaches correlating multiple types of logs together in its system and recognizes the value of so doing. (*See* Ex. 1006, p. 3: "If all this information can be brought together it would enable more robust attack detection mechanisms" and "In the bottom-up approach we try to gather relevant information from multiple logs to identify specific attack instances… Once an anomaly is detected in one of the logs, the next step is to examine other logs in the same time period."). A POSITA would have found Piccard's logs to be compatible with Abad's correlation and data mining techniques.

171. Thus, Abad in view of Piccard discloses limitation 1[e].

6. Limitation 1[f]

172. Limitation 1[f] of the '243 patent reads:

> *reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.*

173. Abad in view of Piccard discloses limitation 1[f] of the ’243 patent.

174. Abad discloses collecting log information for malware analysis at a centralized point, stating:

> Correlation makes detection more effective and provides information needed for future prevention and reaction. The primary goals of attack trace correlation are to reduce false alerts (i.e., false positives) and identify stealthy attacks (i.e., reduce false negatives). To achieve the aforementioned goals, most of the previous log correlation research focuses on collecting logs in the network to a central point and then using data reduction and correlation to detect intrusions. Correlating multiple logs enables an IDS to make a better decision of whether to trigger an alarm or not, if an anomaly is observed…Silvestro [24] in his Master’s Thesis examines real-time correlation and implements a centralized correlation engine which is an expert system using data collection, aggregation and correlation rules.
> (Ex. 1006, p. 3).

175. Thus, Abad teaches transmitting logs to a central point or centralized correlation engine for analysis, which a POSITA would understand to be an externally networked pestware research entity or to at least motivate transmission of identifying information to such an entity for analysis.

176. Piccard discloses a “reporting module” that reports information about the identified malware and its source (e.g., “a Web site from which the file was downloaded”) to a remote entity for further research:

> In particular, the Web site identification module 120 accesses the history log 116 to identify the Web address of the file. In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site. ***The reporting module 122 then reports this information to a remotely-located computer that is included in the computer network 104.*** This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the potential malware is, in fact, malware and whether the Web site is, in fact, a malware distribution site.
> (Ex. 1007, [0015], emphasis added).

177. A POSITA would have understood that the remotely-located computer as described in Piccard operates as an externally networked pestware research entity. This "remotely-located computer" serves the purpose of analyzing the information to determine whether the website is indeed a source of malware and subsequently reporting its findings. The "reporting module" is responsible for transmitting the necessary information to do such analysis.

> The third operation illustrated in FIG. 2 is to report that the Web site is a potential malware distribution site (block 204). In the illustrated embodiment, once the Web site identification module identifies the Web site, ***a reporting module (e.g., the reporting module 122) reports information regarding the Web site to a remotely-located computer that is connected to the protected computer.*** This information can identify the Web site as a potential malware distribution site, such as in terms of the domain name of the Web site. Alternatively, or in conjunction, this information can identify the file as including potential malware, such as in terms of the name of the file or the URL of the file. It is also contemplated that this information can include a representation of the file or can identify suspicious activities related to the file. ***This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the file does, in fact, include malware and whether the Web site is, in fact, a malware distribution site.*** If the Web site is determined to be a malware distribution site, a new or updated set of malware definitions can be generated based on content within the Web site, and the new or updated set of malware definitions can be provided to the protected computer. In addition, content within the Web site can be monitored on a periodic or other basis for new or updated malware. Also, a new or updated list of malware distribution sites can be generated so as to identify the Web site, and the new or updated list of malware distribution sites can be provided to the protected computer.
> (Ex. 1007, [0023], emphasis added).

178. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better

detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings. Furthermore, Abad teaches correlating multiple types of logs together in its system and recognizes the value of so doing. (*See* Ex. 1006, p. 3: "If all this information can be brought together it would enable more robust attack detection mechanisms" and "In the bottom-up approach we try to gather relevant information from multiple logs to identify specific attack instances… Once an anomaly is detected in one of the logs, the next step is to examine other logs in the same time period."). A POSITA would have found Piccard's logs to be compatible with Abad's correlation and data mining techniques.

179. Thus, Abad in view of Piccard discloses limitation 1[f].

180. Thus, I conclude that Abad in view of Piccard renders Claim 1 obvious to a POSITA.

C. Claim 2 is rendered obvious by Abad in view of Piccard

181. Claim 2 of the '243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 2 reads,

> *The method of claim 1, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.*

182. As I explained with respect to limitation 1[e], Piccard discloses a "Web site identification module" that is invoked after a determination of malware by the "malware detection module", which accesses the history log to determine the external source of the object:

> In particular, when a file is accessed and downloaded, the Web browser 114 records a Web address of the file in the history log 116. A Web address typically specifies a location of a file within a Web site. For example, a Web

> address can be a Uniform Resource Identifier ("URI") of a file, ***such as a Uniform Resource Locator ("URL") of the file. It is also contemplated that a Web address can be defined in various other ways, such as using an Internet Protocol ("IP") address or any other identifier of a source of a file.***
> (Ex. 1007, [0013], emphasis added).
>
> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

183. Accordingly, Piccard teaches identifying a website via a URL or an IP address of the file. In more detail, Piccard discloses that the "Web address of the file" can be a URL:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, ***the Web site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file,*** which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021], emphasis added).

184. Further, a POSITA would have understood that a web URL and an I.P. address are commonly convertible identifiers via any domain name system (DNS) server. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

185. Thus, I conclude that Abad in view of Piccard renders Claim 2 obvious to a POSITA.

D. Claim 3 is rendered obvious by Abad in view of Piccard

186. Claim 3 of the '243 patent, which depends from Claim 2, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 3 reads,

> *The method of claim 2, wherein the identifying includes accessing an activity log that includes information that relates the activity to the suspected pestware object.*

187. As I explained with respect to limitation 1[d], Abad discloses monitoring activity logs, such as system call logs:

> Our architecture consists of logging utilities that monitor system activity from different vantage points. For example, APISpy32 [10] monitors the

> system from the kernel level by recording the system calls executed by the system.
> (Ex. 1006, p. 6).
>
> We used APISpy32 [10], with some modification, to intercept system calls. The logged system calls and network traffic on the client machine were then mined to determine if an intrusion had taken place.
> (Ex. 1006, p. 5).

188. A POSITA would have understood that system call logs identify the process that made the system call. A POSITA would have known it to be possible, and would have found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity (e.g., the claimed "suspected pestware object"), and a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs.

189. Abad describes using the collected data to predict the next system call in a system call sequence, in order to determine whether there was an intrusion:

> Then second approach [i.e., in which RIPPER is trained with "normal" traces but not "abnormal" traces] uses a system call sequence to predict the next system call. After processing the training data, RIPPER generates a series of rules. One such rule may be: if system call 1 is Open, then system call 7 should be Closed (with a confidence level of 0.38). After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples. When mining the actual experimental data from both the network traffic log and system call log, the confidence level of the violated rule is adjusted similar to Lee et al. [14]. The averaged score (by the total number of sequences) of the trace is used to decide whether there is an intrusion.
> (Ex. 1006, pp. 5-6).

190. A POSITA would have understood a traced series of events or calls as relating to the activity of a particular object.

191. Thus, I conclude that Abad in view of Piccard renders Claim 3 obvious to a POSITA.

E. Claim 4 is rendered obvious by Abad in view of Piccard

192. Claim 4 of the ’243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 4 reads,

> *The method of claim 1, wherein the recorded history resides in a browser cache, which includes information about files downloaded to the computer and a source of each of the files.*

193. As I explained with reference to limitation 1[e], Piccard, when a malware file is identified, uses its history log to identify the source of the file. Piccard describes the history log as follows:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, the Web site identification module can access the Web browser's history log to ***identify a Web address of the file***. As described previously, the Web address of the file can be a URL of the file, which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where “http://” specifies a communication protocol used to download the file, “www.DomainName” specifies a domain name of the Web site from which the file was downloaded, “/Subdirectory/” specifies a subdirectory within the Web site from which the file was downloaded, and “FileName.html” specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
>
> As another example, the Web site identification module can generate the download history log based on the Web browser's history log. In this example, the Web site identification module can access the Web browser's

> history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0021]-[0022], emphasis added).

194. Piccard continues, emphasizing the performance benefits of having such a history log:

> By including such salient information in the download history log, the Web site identification module ***can accelerate and simplify a search process***. Further acceleration and simplification of the search process can be achieved by filtering out duplicative entries, such as in the event a same version of a file is downloaded multiple times from a Web site. It is also contemplated that the Web site identification module can generate the download history log independently of the Web browser's history log.
> (Ex. 1007, [0022], emphasis added).

195. The ’243 patent specification describes “browser cache” as follows:

> The browser history 240 in this embodiment includes a listing of web sites and respective times that the web sites were visited by the user. The browser cache 242 in this embodiment includes files, cookies, and other objects cached in connection with use of a browser of the protected computer (e.g., Internet Explorer or Mozilla (not shown)). As depicted, the browser cache 242 includes a cache index 243 that includes a listing, which associates the content of the browser cache 242 with URLs and time stamps. As discussed further herein, the ***cache index 243 provides an efficient means for identifying the times objects were retrieved and the URLs that the objects were retrieved from.***
> (Ex. 1001, 6:24-35, emphasis added).

196. A POSITA would have understood browser cache in the context of this claim to correspond to browser history log. Thus, Piccard’s teachings about associating file names of downloaded files with URLs (e.g., a “Web address”) align with the ’243 patent’s teachings about its browser cache, specifically the “cache index 243”, including the efficiency of such an index.

197. I discussed in Section IX.A. why a POSITA would have been motivated to combine Abad with Piccard, and I incorporate that analysis here by reference.

198. Thus, I conclude that Abad in view of Piccard renders Claim 4 obvious to a POSITA.

F. Claim 5 is rendered obvious by Abad in view of Piccard

199. Claim 5 of the '243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Abad or Abad in view of Piccard. Claim 5 reads,

> *The method of claim 1, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

200. As I explained in my analysis of claim limitation 1[e] (which I incorporate herein by reference), Abad discloses accessing "recorded history" that resides in a network log and a system call log in order to identify malware. Specifically, Abad discloses:

> Since it is unlikely that we will ever be able to prevent all intrusions, being able to detect an intrusion is fundamental. ***As long as traffic flow is allowed between an internal network and the Internet,*** the opportunity for an attacker or worm to penetrate will always exist. Despite the preventive steps taken, intrusions that involve new techniques usually succeed. ***During and after an attack, network and system logs are the primary sources of information for detection, response, and forensics, without these logs there is little chance of ever discovering what happened.***
> (Ex. 1006, p. 2, emphasis added).

> Looking at the system call sequence, abnormal behavior may be observed which is not deemed statistically significant to signify an intrusion in a tuned IDS. However, if we examine the corresponding network traffic the traces show an unusual amount of network traffic corresponding to the system call sequence where the user opens the e-mail. Combining this information from the system call sequence and the network traffic, it is possible to identify intrusions that would normally not be identified using just one log.
> (Ex. 1006, p. 5).

201. A POSITA would have understood that the network log of Abad corresponds to an activity log, an event log, or monitoring software logs.

202. As I explained in my analysis of Claim 4 (which I incorporate herein by reference), Piccard discloses accessing "recorded history" that resides in a "browser cache" by extracting such information from the web browser's history log:

> As another example, the Web site identification module can ***generate the download history log based on the Web browser's history log***. In this example, the Web site identification module can access the Web browser's history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0022], emphasis added).

203. Furthermore, Piccard discloses accessing "recorded history" that resides in "a browser history" by accessing the browser's history log directly:

> In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. … Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021]).

204. Moreover, after aggregation and correlation of the download logs and browser history logs with the logs of Abad, a POSITA would have understood that the network access logs of Abad would be enhanced to relate access history with web site history, and the system call logs of Abad would be enhanced to relate file downloads with new or anomalous system calls.

205. I discussed in Section IX.A. why a POSITA would have been motivated to combine Abad with Piccard, and I incorporate that analysis here by reference.

206. Thus, I conclude that Abad or Abad in view of Piccard renders Claim 5 obvious to a POSITA.

G. Claim 6 is rendered obvious by Abad in view of Piccard

207. Each of the limitations of Claim 6 of the ’243 patent is rendered obvious by Abad in view of Piccard as described below.

1. Limitation 6[a]

208. Limitation 6[a] of the ’243 patent reads:

> *A system for identifying a source of activity on a computer that is indicative of pestware including:*

209. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 6 of the ’243 patent is considered limiting, Abad in view of Piccard discloses the preamble of Claim 6.

210. Abad describes its goal as follows:

> Intrusion detection is an important part of networked-systems security protection. Although commercial products exist, finding intrusions has proven to be a difficult task with limitations under current techniques. Therefore, improved techniques are needed. We argue the need for correlating data among different logs to improve intrusion detection systems accuracy. We show how different attacks are reflected in different logs and argue that some attacks are not evident when a single log is analyzed. We present experimental results using anomaly detection for the virus Yaha. Through the use of data mining tools (RIPPER) and correlation among logs we improve the effectiveness of an intrusion detection system while reducing false positives.
> (Ex. 1006, p. 1).

211. Abad also describes a computer system on which its techniques are implemented:

> The experiments were conducted on two interconnected workstations in a laboratory isolated from the Internet. The Windows XP machine had an e-mail client and logging tools installed and the Linux box served as an e-mail server.
> (Ex. 1006, p. 5).

212. Piccard teaches a method for identifying the source of malware by examining web traffic history:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

213. Piccard also discloses a computer system that implements its methods:

> FIG. 1 illustrates a computer system 100 that is implemented in accordance with an embodiment of the invention. The computer system 100 includes at least one protected computer 102, which is connected to a computer network 104 via any wire of wireless transmission channel. In general, the protected computer 102 can be a client computer, a server computer, or any other device with data processing capability. Thus, for example, the protected computer 102 can be a desktop computer, a laptop computer, a handheld computer, a tablet computer, a personal digital assistant, a cellular telephone, a firewall, or a Web server. In the illustrated embodiment, the protected computer 102 is a client computer and includes conventional client computer components, including a Central Processing Unit ("CPU") 108 that is connected to a network connection device 110 and a memory 112.
> (Ex. 1007, [0012]).

214. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

215. Thus, Abad in view of Piccard discloses limitation 6[a].

2. Limitation 6[b]

216. Limitation 6[b] of the ’243 patent reads:

an activity monitor configured to monitor API calls and to store a history of at least a portion of the API calls in an activity log, wherein the activity monitor includes a kernel-mode driver adapted to intercept the API calls;

217. Abad in view of Piccard discloses limitation 6[b] of the ’243 patent.

218. Abad teaches monitoring system calls, such as Windows API calls, at the kernel level: “Our architecture consists of logging utilities that monitor system activity from different vantage points. For example, APISpy32 [10] monitors the system from the kernel level by recording the system calls executed by the system.” (Ex. 1006, p. 6). Further, Abad states “[t]he logged system calls and network traffic on the client machine were then mined to determine if an intrusion had taken place.” (Ex. 1006, p. 5). As I discussed in Section V.B., a POSITA would have understood that Windows system calls, such as those being recorded by a monitor like APISpy32, relate to the claimed “API calls” of this limitation. (*See* Section V.B.). For example, an API call of CreateProcess indicates one process (perhaps a malicious process) attempting to spawn another process, and monitoring calls to that API would allow analysis of such behavior:

> One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually, to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new process, your interceptor will gain control and perform whatever processing is necessary.
> (Ex. 1013, p. 1).

219. Thus, Abad discloses limitation 6[b].

3. Limitation 6[c]

220. Limitation 6[c] of the ’243 patent reads:

a heuristics module configured to identify an activity on the computer that is indicative of pestware residing on the computer and to analyze the activity to determine whether one or more weighted factors associated with the activity exceeds a threshold; and

221. Abad discloses limitation 6[c] of the ’243 patent.

222. Abad, during its training phase, creates rules that predict the next expected system call in each sequence of system calls:

> RIPPER expects a list of system calls followed by the value that it will predict. For example, if we decide that we are going to use five system calls to predict the sixth, then RIPPER expects the five system calls followed by the sixth that it will try to predict. A sliding window was used to scan the normal traces and create a list of unique sequences of system calls.
> (Ex. 1006, p. 6).

223. Each rule also has an associated confidence level that indicates how widely the rule applies among the training data:

> After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples.
> (Ex. 1006, p. 5).

224. Using these rules and confidence levels, Abad teaches using monitored activity to determine whether the activity is anomalous (e.g., “abnormal”).

> Approximately 20% of the captured data was saved to use for testing only; i.e., it was not used in training. The generated rules were applied to the testing data to determine how abnormal the traces were. RIPPER returned a list that consists of each predicted result (predicted based upon the rules) along with the actual value. It also gives the confidence level for the particular rule that was used in the prediction.
> (Ex. 1006, p. 6).

225. Abad compares actual observed system call sequences with the rules generated by RIPPER (i.e., the "predicted values") and calculates penalty values which, when summed and compared with a threshold, represent how normal or abnormal a sequence was:

> A sliding window of size 13 was passed over the list of predicted values and actual values returned by RIPPER. If the predicted value differs from the actual value, the penalty value is set to the confidence level of the rule that was broken. Then the penalty values are summed from the 13 spaces of the sliding window and if the value is greater than half (6.5), this "region" is considered as abnormal, otherwise it is considered as normal.
> (Ex. 1006, p. 6).

226. Abad teaches that having weighted factors (e.g., "confidence level" of rules) is desirable, to increase the likelihood of correct classification:

> The reason we chose to use the confidence level as a penalty value is to avoid rules having equal weight. For example, a rule that has a confidence level of 50% should not have the same penalty cost as a rule with confidence level of 99%.
> (Ex. 1006, p. 7).

227. In addition, Abad teaches heuristically applying thresholds when correlating activity from system calls with activity from network traffic, to determine whether the overall activity during a given time period is "abnormal". For example, the threshold value of 6.5 cited earlier may be increased or decreased based on some of the activity:

> Figure 1 illustrates the correlation process. Figure 1(a) shows how the sliding window is used to detect if a region is abnormal. In Figure 1(a) the region is normal, since there are only 3 abnormals and the threshold is 3.5. In Figure 1(b) we lower the threshold by 1. Notice that the same region is now considered abnormal. Lowering the threshold is equivalent to increasing suspicion.
>
> The results from the previous step were used in processing the system calls again, by following the same steps in Section 8.6 except that the time sequences that were determined were used along with their penalty values in correlating the data. The penalty value was used to adjust the threshold to determine if a region is "abnormal".

Then the system calls were processed. ***When there was a sequence with a time value between 1pm and 1:10pm the threshold was lowered by the penalty value (3). This has the effect of increasing the suspicion level based on the data in a different log. After the correlation, it only took a 3.5 value to declare a region as abnormal as opposed to the original 6.5 value.***

We also decided to increase the threshold during normal time periods. We expect this to help to eliminate false positives. ***For example, if network traffic is normal between 1:10pm and 1:30pm we can increase the threshold to detect an abnormal region by 1. So, it would take 7.5 to label a region as abnormal during that time period.***
(Ex. 1006, p. 7, emphasis added).

228. Figure 1 illustrates the effect of the threshold on the outcome:

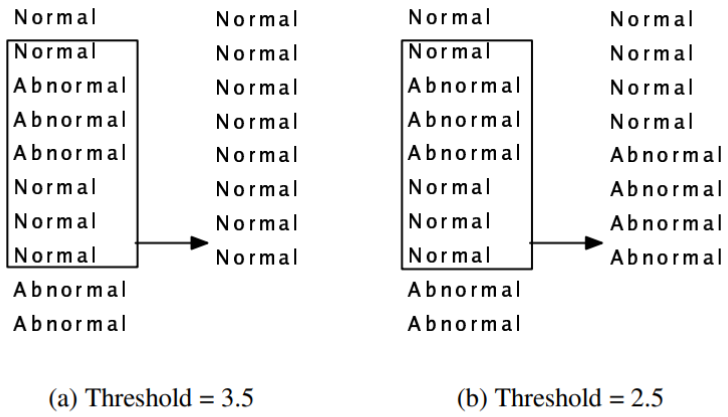


Figure 1. Sliding window in the correlation process

(Ex. 1006, p. 7, Figure 1).

229. Thus, Abad discloses limitation 6[c].

4. Limitation 6[d]

230. Limitation 6[d] of the ’243 patent reads:

a research portion configured to access, in response to a prompt from the heuristics module, a first set of recorded information on the computer that relates the activity to at least one file residing on the computer, and wherein the research portion is configured to access a second set of recorded

information on the computer that relates the at least one file to an externally networked source from which the file was received; and

231. Abad in view of Piccard discloses limitation 6[d] of the '243 patent.

232. Abad describes using the collected system call logs to determine whether there was an intrusion:

> Then second approach [i.e., in which RIPPER is trained with "normal" traces but not "abnormal" traces] uses a system call sequence to predict the next system call. After processing the training data, RIPPER generates a series of rules. One such rule may be: if system call 1 is Open, then system call 7 should be Closed (with a confidence level of 0.38). After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples. When mining the actual experimental data from both the network traffic log and system call log, the confidence level of the violated rule is adjusted similar to Lee et al. [14]. The averaged score (by the total number of sequences) of the trace is used to decide whether there is an intrusion.
> (Ex. 1006, pp. 5-6).

233. A POSITA would have understood that system call logs identify the process that made the system call. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity, as a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs, and the information associating the system call activity to the file would represent the claimed "a first set of recorded information on the computer that relates the activity to at least one file residing on the computer".

234. Abad also describes collecting network logs and network activity / traffic to assist in tracing intrusions. (*See* Ex. 1006, pp. 3, 5). After aggregation and correlation of the download

logs and browser history logs of Piccard with the logs of Abad, a POSITA would have understood that the network access logs of Abad would be enhanced to relate access history with web site history, and the system call logs of Abad would be enhanced to relate file downloads with new or anomalous system calls.

235. Piccard teaches that its "Web site identification module" starts its search of the history log after determination by the "malware detection module" that a file includes potential malware:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, ***once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded.*** The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file, which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021], emphasis added).

236. The purpose of Piccard's "Web site identification module" is to associate the suspicious file with its URL source. Thus, Piccard's "history log" teaches the claimed "a second set of recorded information on the computer that relates the at least one file to an externally networked source from which the file was received", which is accessed "in response to a prompt from the heuristics module" (e.g., the "malware detection module").

237. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings. Furthermore, Abad teaches correlating multiple types of logs together in its system and recognizes the value of so doing. (*See* Ex. 1006, p. 3: "If all this information can be brought together it would enable more robust attack detection mechanisms" and "In the bottom-up approach we try to gather relevant information from multiple logs to identify specific attack instances… Once an anomaly is detected in one of the logs, the next step is to examine other logs in the same time period."). A POSITA would have found Piccard's logs to be compatible with Abad's correlation and data mining techniques.

238. Thus, Abad in view of Piccard discloses limitation 6[d].

5. Limitation 6[e]

239. Limitation 6[e] of the '243 patent reads:

> *a reporting portion configured to generate a report that identifies the externally networked source of the file and to report an identity of the externally networked source to an externally networked pestware research*

entity so as to enable the externally networked pestware research entity to research whether the externally networked source is a source of pestware.

240. Abad in view of Piccard discloses limitation 6[e] of the '243 patent.

241. Piccard discloses a "reporting module" that reports information about the identified malware and its source (e.g., "a Web site from which the file was downloaded") to a remote entity for further research:

> In particular, the Web site identification module 120 accesses the history log 116 to identify the Web address of the file. In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site. ***The reporting module 122 then reports this information to a remotely-located computer that is included in the computer network 104.*** This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the potential malware is, in fact, malware and whether the Web site is, in fact, a malware distribution site.
> (Ex. 1007, [0015], emphasis added).

242. A POSITA would have understood the remotely-located computer as described in Piccard operates as an externally networked pestware research entity. This "remotely-located computer" serves the purpose of analyzing the information to determine whether the website is indeed a source of malware and subsequently reporting its findings. The "reporting module" is responsible for transmitting the necessary information to do such analysis.

> The third operation illustrated in FIG. 2 is to report that the Web site is a potential malware distribution site (block 204). In the illustrated embodiment, once the Web site identification module identifies the Web site, ***a reporting module (e.g., the reporting module 122) reports information regarding the Web site to a remotely-located computer that is connected to the protected computer.*** This information can identify the Web site as a potential malware distribution site, such as in terms of the domain name of the Web site. Alternatively, or in conjunction, this information can identify the file as including potential malware, such as in terms of the name of the file or the URL of the file. It is also contemplated that this information can include a representation of the file or can identify suspicious activities related to the file. ***This information as well as any***

> ***additional relevant information can be analyzed at the remotely-located computer to determine whether the file does, in fact, include malware and whether the Web site is, in fact, a malware distribution site.*** If the Web site is determined to be a malware distribution site, a new or updated set of malware definitions can be generated based on content within the Web site, and the new or updated set of malware definitions can be provided to the protected computer. In addition, content within the Web site can be monitored on a periodic or other basis for new or updated malware. Also, a new or updated list of malware distribution sites can be generated so as to identify the Web site, and the new or updated list of malware distribution sites can be provided to the protected computer.
> (Ex. 1007, [0023], emphasis added).

243. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings. Furthermore, Abad teaches correlating multiple types of logs together in its system and recognizes the value of so doing. (*See* Ex. 1006, p. 3: "If all this information can be brought together it would enable more robust attack detection mechanisms" and "In the bottom-up approach we try to gather relevant information from multiple logs to identify specific attack instances… Once an anomaly is detected in one of the logs, the next step is to examine other logs in the same time period."). A POSITA would have found Piccard's logs to be compatible with Abad's correlation and data mining techniques.

244. Thus, Abad in view of Piccard discloses limitation 6[e].

245. Thus, I conclude that Abad in view of Piccard renders Claim 6 obvious to a POSITA.

H. Claim 7 is rendered obvious by Abad in view of Piccard

246. Claim 7 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 7 reads,

> *The system of claim 6, wherein activity monitor is configured to store an identity of each process that made each of the API calls in the stored history, and wherein the activity monitor is configured to store an identity of each file that corresponds to each the processes that made the API calls in the stored history so as to create a relation between each API call and at least one file.*

247. In the combined Abad-Piccard system, in which a POSITA is motivated to identify the source of the suspected malicious file, it would have been obvious to both obtain the identity of the file (e.g., through known techniques detailed in Sections V.A. and V.C.) and to store that identity for future reference, in order to gain processing efficiency by acquiring the information only once. Abad correlating logs to identify links between activity such as network activity and system call activity, and in the combined Abad-Piccard system this network activity would include files downloaded from the Internet. (Ex. 1006, p. 5-6). Thereafter, since the stored activity includes the API call and the process identity associated with it, it would have been obvious to a POSITA that the identity of the file corresponding to the process can be trivially obtained (e.g., by storing it alongside the activity in the log, or by looking up the stored file name based on the process ID stored in the activity log). After aggregation and correlation of the download logs and browser history logs of Piccard with the logs of Abad, a POSITA would have understood that the network access logs of Abad would be enhanced to relate access history with web site history, and

the system call logs of Abad would be enhanced to relate file downloads with new or anomalous system calls. Furthermore, Abad teaches correlating multiple types of logs together in its system and recognizes the value of so doing. (*See* Ex. 1006, p. 3: "If all this information can be brought together it would enable more robust attack detection mechanisms" and "In the bottom-up approach we try to gather relevant information from multiple logs to identify specific attack instances… Once an anomaly is detected in one of the logs, the next step is to examine other logs in the same time period."). A POSITA would have found Piccard's logs to be compatible with Abad's correlation and data mining techniques.

248. Thus, I conclude that Abad in view of Piccard renders Claim 7 obvious to a POSITA.

I. Claim 8 is rendered obvious by Abad in view of Piccard

249. Claim 8 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 8 reads,

> *The system of claim 6, wherein the heuristics module is configured to receive information about the API calls from the activity monitor and determine whether the API calls are indicative of pestware.*

250. Abad discloses using system calls to identify abnormal activity. A POSITA would have understood that system calls are API calls. (*See* Section V.B.).

251. Abad, during its training phase, creates rules that predict the next expected system call in each sequence of system calls:

> RIPPER expects a list of system calls followed by the value that it will predict. For example, if we decide that we are going to use five system calls to predict the sixth, then RIPPER expects the five system calls followed by the sixth that it will try to predict. A sliding window was used to scan the normal traces and create a list of unique sequences of system calls.
> (Ex. 1006, p. 6).

252. Each rule also has an associated confidence level that indicates how widely the rule applies among the training data:

> After RIPPER generates the rules, the confidence level of each rule is calculated as the number of matched examples divided by the sum of matched and unmatched examples.
> (Ex. 1006, p. 5).

253. Using these rules and confidence levels, Abad teaches using monitored activity to determine whether the activity is anomalous (e.g., "abnormal").

> Approximately 20% of the captured data was saved to use for testing only; i.e., it was not used in training. The generated rules were applied to the testing data to determine how abnormal the traces were. RIPPER returned a list that consists of each predicted result (predicted based upon the rules) along with the actual value. It also gives the confidence level for the particular rule that was used in the prediction.
> (Ex. 1006, p. 6).

254. Thereafter, Abad compares the actual observed system call sequences with the rules generated by RIPPER (i.e., the "predicted values") and calculates penalty values which, when summed and compared with a threshold, represent how normal or abnormal a sequence was:

> A sliding window of size 13 was passed over the list of predicted values and actual values returned by RIPPER. If the predicted value differs from the actual value, the penalty value is set to the confidence level of the rule that was broken. Then the penalty values are summed from the 13 spaces of the sliding window and if the value is greater than half (6.5), this "region" is considered as abnormal, otherwise it is considered as normal.
> (Ex. 1006, p. 6).

255. Thus, I conclude that Abad in view of Piccard renders Claim 8 obvious to a POSITA.

J. Claim 9 is rendered obvious by Abad in view of Piccard

256. Claim 9 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 9 reads,

The system of claim 6, wherein the first set of recorded information and the second set of recorded information reside in the same log file on a file storage device of the computer.

257. As I discussed in Section V.A., a POSITA would have understood the value of storing various sets of logging, auditing, or security information (e.g., the claimed "recorded information") within a common file or database on a computer, and it was commonly practiced in the art. (*See* Section V.A.; *see also* Ex. 1010, pp. 15-17). Further, Abad explicitly teaches such data aggregation and storage in a single log file for data mining. (Ex. 1006, pp. 3, 5-6).

258. As I discussed with respect to limitation 6[d], the "recorded information" stored and accessed by Abad was stored on the computer, and the "recorded information" stored and accessed by Piccard was also stored on the computer. For example, Abad describes how activity log aggregation was well practiced in the industry such that data was hosted at a central point. (*See* Ex. 1006, p. 3). A POSITA implementing the combination would have been motivated by the benefits enumerated by Rinnan to store these sets of information together, perhaps in a normalized form such as "a process that breaks an event into its component parts and places them into individual database fields." (Ex. 1010, p. 16).

259. A POSITA would have understood that a database, in order to preserve its information long-term, is stored in a file on a file storage device.

260. Thus, I conclude that Abad in view of Piccard renders Claim 9 obvious to a POSITA.

K. Claim 10 is rendered obvious by Abad in view of Piccard

261. Claim 10 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 10 reads,

The system of claim 6, wherein the externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.

262. As I explained with respect to limitation 6[d], Piccard discloses a "Web site identification module" that is invoked after a determination of malware by the "malware detection module", which accesses the history log to determine the external source of the object:

> In particular, when a file is accessed and downloaded, the Web browser 114 records a Web address of the file in the history log 116. A Web address typically specifies a location of a file within a Web site. For example, a Web address can be a Uniform Resource Identifier ("URI") of a file, ***such as a Uniform Resource Locator ("URL") of the file. It is also contemplated that a Web address can be defined in various other ways, such as using an Internet Protocol ("IP") address or any other identifier of a source of a file.***
> (Ex. 1007, [0013], emphasis added).
>
> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

263. Accordingly, Piccard teaches identifying a website via a URL or an IP address of the file. In more detail, Piccard discloses that the "Web address of the file" can be a URL:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, ***the Web***

> ***site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file,*** which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021], emphasis added).

264. Further, a POSITA would have understood that a web URL and an I.P. address are commonly convertible identifiers via any domain name system (DNS) server. I discussed in Section IX.A. why a POSITA would have been motivated to combine Abad with Piccard, and I incorporate that analysis here by reference.

265. Thus, I conclude that Abad in view of Piccard renders Claim 10 obvious to a POSITA.

L. Claim 11 is rendered obvious by Abad in view of Piccard

266. Claim 11 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 11 reads,

> *The system of claim 6 wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

267. As I explained in my analysis of Claim 10 (which I incorporate herein by reference), Piccard discloses accessing "recorded history" that resides in a "browser cache" by extracting such information from the web browser's history log:

> As another example, the Web site identification module can ***generate the download history log based on the Web browser's history log***. In this example, the Web site identification module can access the Web browser's history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0022], emphasis added).

268. Furthermore, Piccard discloses accessing "recorded history" that resides in "a browser history" by accessing the browser's history log directly:

> In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. … Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021]).

269. Moreover, Abad discloses correlating network traffic logs, which a POSITA would have understood to represent one type of "activity log", as they contain records of network activity between the computer and its internal or external networks:

> Since it is unlikely that we will ever be able to prevent all intrusions, being able to detect an intrusion is fundamental. ***As long as traffic flow is allowed between an internal network and the Internet,*** the opportunity for an attacker or worm to penetrate will always exist. Despite the preventive steps taken, intrusions that involve new techniques usually succeed. ***During and after an attack, network and system logs are the primary sources of information for detection, response, and forensics, without these logs there is little chance of ever discovering what happened.***
> (Ex. 1006, p. 2, emphasis added).

> Looking at the system call sequence, abnormal behavior may be observed which is not deemed statistically significant to signify an intrusion in a tuned IDS. However, if we examine the corresponding network traffic the traces show an unusual amount of network traffic corresponding to the system call sequence where the user opens the e-mail. Combining this information from the system call sequence and the network traffic, it is possible to identify intrusions that would normally not be identified using just one log.
> (Ex. 1006, p. 5).

270. I discussed in Section IX.A. why a POSITA would have been motivated to combine Abad with Piccard, and I incorporate that analysis here by reference.

271. Thus, I conclude that Abad in view of Piccard renders Claim 11 obvious to a POSITA.

M. Claim 12 is rendered obvious by Abad in view of Piccard

272. Each of the limitations of Claim 12 of the ’243 patent is rendered obvious by Abad in view of Piccard as described below.

1. Limitation 12[a]

273. Limitation 12[a] of the ’243 patent reads:

> *A non-transitory computer-readable medium including processor-executable instructions for identifying an origin of activity on a computer that is indicative of pestware, the instructions including instructions for:*

274. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 12 of the ’243 patent is considered limiting, Abad in view of Piccard discloses the preamble of Claim 12.

275. Abad describes its goal as follows:

> Intrusion detection is an important part of networked-systems security protection. Although commercial products exist, finding intrusions has proven to be a difficult task with limitations under current techniques. Therefore, improved techniques are needed. We argue the need for correlating data among different logs to improve intrusion detection systems accuracy. We show how different attacks are reflected in different logs and argue that some attacks are not evident when a single log is analyzed. We present experimental results using anomaly detection for the virus Yaha. Through the use of data mining tools (RIPPER) and correlation among logs we improve the effectiveness of an intrusion detection system while reducing false positives.
> (Ex. 1006, p. 1).

276. Abad also describes a computer system on which its techniques are implemented:

> The experiments were conducted on two interconnected workstations in a laboratory isolated from the Internet. The Windows XP machine had an e-mail client and logging tools installed and the Linux box served as an e-mail server.
> (Ex. 1006, p. 5).

> Our setup consisted of 2 Dell Workstations with 2.56 GHz Pentium 4 processors and 512 MB RAM.
> (Ex. 1006, p. 6).

277. Piccard teaches a method for identifying the source of malware by examining web traffic history:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

278. Piccard also discloses a computer system that implements its methods:

> FIG. 1 illustrates a computer system 100 that is implemented in accordance with an embodiment of the invention. The computer system 100 includes at least one protected computer 102, which is connected to a computer network 104 via any wire of wireless transmission channel. In general, the protected computer 102 can be a client computer, a server computer, or any other device with data processing capability. Thus, for example, the protected computer 102 can be a desktop computer, a laptop computer, a handheld computer, a tablet computer, a personal digital assistant, a cellular telephone, a firewall, or a Web server. In the illustrated embodiment, the protected computer 102 is a client computer and includes conventional client computer components, including a Central Processing Unit ("CPU") 108 that is connected to a network connection device 110 and a memory 112.
> (Ex. 1007, [0012]).

> In the illustrated embodiment, the memory 112 also stores a set of computer programs that implement the operations described herein. In particular, the memory 112 stores a malware detection module 118, a Web site identification module 120, a reporting module 122, and a malware removal module 124. As further described below, the various modules 118, 120, 122, and 124 operate to manage malware that can be present in the computer system 100.

(Ex. 1007, [0014]).

279. As I discussed in Section IX.A., a POSITA would have been motivated to combine Abad with Piccard. Abad is directed to the identification of pestware and recognizes the efficiency and accuracy gained through correlating multiple sources of activity information in that identification. Abad further teaches the need to correlate information on a network level, a concept taught by Piccard's identification and reporting modules. Piccard teaches identifying the origin of pestware (e.g., a URL) and sharing that information with a research entity in order to provide better detection capabilities to nodes on the network. Furthermore, Piccard teaches a "malware detection module" with various exemplary techniques for detecting malware. Abad's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

280. Thus, Abad in view of Piccard discloses limitation 12[a].

2. Limitation 12[b]-12[f]

281. The remaining limitations of Claim 12 are virtually identical to the corresponding limitations of Claim 1. I have summarized the differences in the following table:

1[b]: monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;	**12[b]:** monitoring, **with** a kernel-mode driver, the computer for activity that is indicative of pestware, wherein **the instructions for** monitoring include **instructions for** monitoring API calls and storing a history of at least a portion of the API calls in an activity log;

1[c]: analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;	**12[c]:** analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;
1[d]: identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;	**12[d]:** identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;
1[e]: accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and	**12[e]:** accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and
1[f]: reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the	**12[f]:** reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the

particular externally networked source is a source of pestware.	particular externally networked source is a source of pestware.

282. I incorporate herein by reference my analyses of limitations 1[b]-1[f] and conclude that Abad in view of Piccard renders Claim 12 obvious to a POSITA.

N. Claim 13 is rendered obvious by Abad in view of Piccard

283. Claim 13 of the ’243 patent, which depends from Claim 12, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 13 reads,

> *The non-transitory computer-readable medium of claim 12, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.*

284. This claim is virtually identical to Claim 2. I incorporate my analysis of Claim 2 herein by reference and conclude that Abad in view of Piccard renders Claim 13 obvious to a POSITA.

O. Claim 14 is rendered obvious by Abad in view of Piccard

285. Claim 14 of the ’243 patent, which depends from Claim 12, is rendered obvious to a POSITA by Abad in view of Piccard. Claim 14 reads,

> *The non-transitory computer-readable medium of claim 12, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

286. This claim is virtually identical to Claim 5. I incorporate my analysis of Claim 5 herein by reference and conclude that Abad in view of Piccard renders Claim 14 obvious to a POSITA.

X. GROUND 2: CLAIMS 1-14 ARE RENDERED OBVIOUS BY KILPATRICK IN VIEW OF PICCARD

287. As I describe in greater detail hereinbelow, a POSITA would have found Claims 1-14 rendered obvious by Kilpatrick in view of Piccard.

A. Motivation to Combine Kilpatrick and Piccard

288. A POSITA would have been motivated to combine the teachings of Kilpatrick and Piccard. As I discussed in Section VIII.C., Kilpatrick is directed toward detecting anomalous behavior through monitoring system calls and comparing them with a program-specific distance matrix:

> The distance matrix indirectly specifies known sequences by specifying allowable separation distances between pairs of system calls. The distance matrix is used to determine whether a sequence of system calls in an event window represents an anomaly. Anomalies that are detected are further analyzed through levenshtein distance calculations that also rely on the contents of the distance matrix.
> (Ex. 1008, 3:10-17).

289. Kilpatrick discloses the use of wrapper specifications (i.e., criteria to specify when a wrapper should be loaded) that anticipate particular directories being useful targets for monitoring:

> In general, a system can have many different WSs loaded, with different activation criteria. For example, a WS might apply to all programs run by a specific user, or to all instances of programs run from a particular directory. As illustrated in FIG. 2, because the interception occurs inside the kernel, the wrappers cannot be bypassed by malicious code.
> (Ex. 1008, 4:56-62).

290. Kilpatrick also anticipates that various actions may be taken to mitigate the impact of malicious code once it is detected:

> Based upon the value of the minimum levenstein distance, the intrusion detection system can then determine whether an intrusion alert, or other forms of intrusion detection processing, should be initiated. For example, if the minimum levenstein distance is greater than a predefined threshold, the intrusion detection system can initiate steps to terminate the abnormal process.
> (Ex. 1008, 11:22-28).

291. Kilpatrick also describes logging system call activity based on the directory that the executable file is located in. (Ex. 1008, 4:55-65). A POSITA would have understood that the executable program logs of Kilpatrick would relate the system call activity to programs in directories and that the system call logs of Kilpatrick would be combined with the logs of Piccard to relate file downloads with new or anomalous system calls.

292. A POSITA would have understood that locations of files downloaded from external sources would be likely to have higher a probability of containing malicious code than other locations and would have been motivated to seek a reference that teaches appropriate intrusion detection actions to take on malicious code found in such locations.

293. Piccard teaches steps to take after identification of malware downloaded from an external source:

> Techniques are currently available to detect and remove malware. But as malware evolves, techniques for detecting and removing malware should also evolve. Accordingly, current techniques for detecting and removing malware are not always satisfactory and will likely not be satisfactory in the future. Current techniques for detecting and removing malware often use definitions of known malware to scan files of a protected computer. However, it is often difficult to initially locate malware in order to generate the definitions, particularly since malware can evolve. ***In particular, it would be desirable to identify sources of malware, such that definitions can be generated or updated to account for evolving malware.*** In addition, identification of sources of malware would allow a blacklist of Web sites to be generated.
> (Ex. 1007, [003], emphasis added).

294. Piccard teaches a "malware detection module" as the first operation in its method, "to analyze a file to determine that the file includes potential malware (block 200)." (Ex. 1007, [0019]). Several possible techniques for such detection are disclosed, including monitoring for suspicious activity patterns:

> In the illustrated embodiment, the malware detection module compares the file with a set of malware definitions to determine if the file includes potential malware. The set of malware definitions can include representations of malware, suspicious activities that are indicative of or that are common to malware, or both. For example, the set of malware definitions can include a hash value or a digital signature of malware. Such as one that is generated using Message Digest 5 ("MD5"). In this example, the malware detection module generates a hash value for the file and compares the hash value of the file with a set of hash values of malware to determine whether there is a sufficient match. As another example, the set of malware definitions can include a Cyclical Redundancy Code ("CRC") of a portion of malware. In this example, the malware detection module generates a CRC for the file and compares the CRC of the file with a set of CRCs of malware to determine whether there is a sufficient match. ***As a further example, the set of malware definitions can include suspicious activities related to third-party cookies or related to entries or modifications of registry files of an operating system.***
> (Ex. 1007, [0020], emphasis added).

295. A POSITA would have recognized that Kilpatrick's teachings of detecting suspicious behavior through system call monitoring align with Piccard's disclosures of its "malware detection module". Because Piccard discloses the "malware detection module" as a discrete component with multiple possible implementations but a defined output (i.e., the suspected file), a POSITA would have expected a high degree of success implementing Kilpatrick's techniques as part of Piccard's malware detection module.

B. Claim 1 is rendered obvious by Kilpatrick in view of Piccard

296. Each of the limitations of Claim 1 of the '243 patent is rendered obvious by Kilpatrick in view of Piccard as described below.

1. Limitation 1[a]

297. Limitation 1[a] of the ’243 patent reads:

A method for identifying an origin of activity on a computer that is indicative of pestware comprising:

298. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 1 of the ’243 patent is considered limiting, Kilpatrick in view of Piccard discloses the preamble of Claim 1.

299. Kilpatrick discloses a method of intrusion detection that monitors applications for anomalous behaviors:

> The present invention relates generally to intrusion detection, and more particularly, to systems and methods for implementing real-time sequence-based anomaly detection.
> (Ex. 1008, 1:9-11).

300. Kilpatrick explains the purpose of intrusion detection as follows:

> The second line of defense of the network security plan is intrusion detection. After an intruder has successfully penetrated the perimeter and gained access to systems within the protected network, a mechanism must exist to quickly detect the intruder and minimize the damage that the intruder can inflict.
> (Ex. 1008, 1:21-27).

> A real-time sequence-based anomaly detection system is disclosed. In a preferred embodiment, the intrusion detection system is incorporated as part of a software wrapper. Event abstraction in the software wrapper enables the intrusion detection system to apply generically across various computing platforms. Real-time anomaly detection is enabled through the definition of a distance matrix that defines allowable separation distances between pairs of system calls.
> (Ex. 1008, Abstract).

> In accordance with the present invention, a software wrapper provides a dynamically installable, non-bypassable, operating system extension that intercepts a selected program’s system calls. For each system call it intercepts, a wrapper may perform functions such as s (1) add functionality (intrusion detection, auditing, encryption) to the system call, (2) transform the system call (e.g. by replacing it with another), or (3) prevent the system call’s execution (e.g., perform access control).

(Ex. 1008, 4:17-25).

301. Piccard also teaches a method for identifying the source of malware by examining web traffic history: "Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded." (Ex. 1007, [0005]).

302. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. Kilpatrick is directed to the identification of processes exhibiting abnormal behavior and anticipates the need for appropriate responses to such processes. Piccard teaches identifying the origin of pestware that has been detected by techniques like those taught in Kilpatrick. Kilpatrick's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

303. Thus, Kilpatrick in view of Piccard discloses limitation 1[a].

2. Limitation 1[b]

304. Limitation 1[b] of the '243 patent reads:

> *monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;*

305. Kilpatrick discloses limitation 1[b] of the '243 patent.

306. Kilpatrick discloses a "kernel-loadable module" that monitors API calls made by an application:

> As illustrated in FIG. 2, operating within the UNIX environment are programs 202A and 202B. Program 202B is an unwrapped program whose system calls pass directly to an internal application programming interface (API). Program 202A, on the other hand, is "wrapped' by software wrapper 210. As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210 based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).
>
> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. These wrapper instances wrap their processes by intercepting some or all of the system calls that are made by the process. This interception effectively puts the wrappers in complete control of their processes' interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:33-55).

307. Kilpatrick is explicit that the monitoring occurs in the kernel:

> As illustrated in FIG. 2, because the interception occurs inside the kernel, the wrappers cannot be bypassed by malicious code.
> (Ex. 1008, 4:59-62).

308. As I discussed in Section V.B., a POSITA would have understood that Windows system calls, such as those being recorded by a kernel-mode driver, relate to the claimed "API calls" of this limitation. (*See* Section V.B.). For example, an API call of CreateProcess indicates one process (perhaps a malicious process) attempting to spawn another process, and monitoring calls to that API would allow analysis of such behavior:

> One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually, to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new

> process, your interceptor will gain control and perform whatever processing is necessary.
> (Ex. 1013, p. 1).

309. Kilpatrick discloses storing the monitored API calls into a rolling "event window", which is later used for analysis:

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
> (Ex. 1008, 8:33-38).

310. Thus, Kilpatrick discloses limitation 1[b].

3. Limitation 1[c]

311. Limitation 1[c] of the '243 patent reads:

> *analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;*

312. Kilpatrick discloses limitation 1[c] of the '243 patent.

313. After the system calls are stored in the event window, they can be analyzed by comparing them with a distance matrix created from known-good training data:

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
>
> At this point, the system determines whether the new sequence of system calls in event window 300 is a recognized sequence. In accordance with the present invention, the determination process is not based upon a literal comparison of entire sequences of system calls. Rather, multiple pairs of system calls are compared relative to the predefined contents of distance matrix 220.
> (Ex. 1008, 8:33-45).

314. If such comparison reveals that the system call sequence exists in the distance matrix, no further processing is necessary:

> If each system call pair has been determined to exist in distance matrix 220, then the sequence of system calls that resides in event window 300 represents a known sequence of system calls. The process then loops back to step 502 where the next system call is received.
> (Ex. 1008, 9:3-8).

315. However, if the sequence was not found, Kilpatrick teaches performing anomaly analysis on the sequence by measuring the magnitude of the difference between known behavior (represented in the distance matrix) and the observed unknown/potentially abnormal behavior:

> If, at step 508, one of the system call pairs has been determined not to exist in distance matrix 220, then the system call sequence in event window 300 has not been seen before. In this case, the detect process proceeds to step 510, where anomaly analysis processing is performed***. In the anomaly analysis processing step, the magnitude of the anomaly is identified. This magnitude measure represents the extent to which the system call sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220.*** In general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion.
>
> In a preferred embodiment of the present invention, a levenshtein distance is used as a measure of the magnitude of the difference between a system call sequence in a stored profile and the system call sequence that is generated by a program.
> (Ex. 1008, 9:8-23, emphasis added).

316. In other words, Kilpatrick discloses comparing a new system call sequence to other collected system calls in the distance matrix (Ex. 1008, 9:7-66) using a Levenshtein distance between each system call pair. These distances are then combined into a single magnitude of the difference which still "represents the extent to which the system call sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220." A POSITA would have understood that these distances are weighted to combine to a single magnitude of the

difference. (*See* Ex. 1008, Equation 1, 9:60-65). Kilpatrick then discloses that once the "magnitude" or "distance" calculations are complete, the system can determine that the process is abnormal if the distance exceeds a threshold:

> The minimum value across all possible values of test head therefore represents the minimum levenshtein distance across all known sequences of system calls. Based upon the value of the minimum levenstein distance, the intrusion detection system can then determine whether an intrusion alert, or other forms of intrusion detection processing, should be initiated. For example, if the minimum levenstein distance is greater than a predefined threshold, the intrusion detection system can initiate steps to terminate the abnormal process.
> (Ex. 1008, 11:19-28).

317. Further, as described above, Kilpatrick also discloses that "in general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion", which a POSITA would have understood to weight variations as an indicator of intrusion activity and determine a significant likelihood of intrusion. A POSITA would have understood comparing a number of distance values against a predefined threshold by using Kilpatrick's "minimum value" involves weighting the other values. (Ex. 1008, 11:19-28).

318. Thus, Kilpatrick discloses limitation 1[c].

4. Limitation 1[d]

319. Limitation 1[d] of the '243 patent reads:

> *identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;*

320. Kilpatrick discloses limitation 1[d] of the '243 patent.

321. Kilpatrick discloses monitoring activity of individual programs, enabling it to identify the object that caused the activity:

> As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210

> based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).
>
> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. These wrapper instances wrap their processes by intercepting Some or all of the System calls that are made by the process. This interception effectively puts the wrappers in complete control of their processes interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:37-55).
>
> The intrusion detection system therefore runs in two modes: record mode and detect mode. In record mode, the intrusion detection system generates a normative sequence database for each program executed. A per-program database for every program executed is thereby created.
> (Ex. 1008, 5:54-58).

322. Kilpatrick teaches monitoring individual programs whose identities are known to the wrapper doing the monitoring. Kilpatrick also describes logging system call activity based on the directory that the executable file is located in. (Ex. 1008, 4:55-65). A POSITA would have understood that the executable program logs of Kilpatrick would relate the system call activity to programs in directories and that the system call logs of Kilpatrick would be combined with the logs of Piccard to relate file downloads with new or anomalous system calls.

323. In addition, a POSITA would have understood that system call logs, such as those taught by Kilpatrick, identify the process that made the system call. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of

the activity (e.g., the claimed "suspected pestware object"), and a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs.

324. Thus, Kilpatrick discloses limitation 1[d].

5. Limitation 1[e]

325. Limitation 1[e] of the '243 patent reads:

> *accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and*

326. Kilpatrick in view of Piccard discloses limitation 1[e] of the '243 patent.

327. Piccard teaches this limitation in its disclosures about two of the modules in its system. One, a "malware detection module," performs the claimed "identifying an object". The malware detection module of Piccard analyzes files that have been downloaded and, in the combined Kilpatrick-Piccard system, analyzes files identified as the sources of anomalous system call sequences as described regarding claim limitations 1[c] and 1[d]. Thereafter, Piccard's "Web site identification module" attempts to identify the external source of the malware:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

328. In more detail, Piccard discloses that the "Web site identification module" is invoked after a determination of malware by the "malware detection module", and that it accesses the history log to determine the external source:

> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

329. Piccard's "history log" teaches the claimed "a recorded history of externally networked sources that the computer received files from", and "a Web address from which the file was received" teaches the claimed "a reference to an identity of a particular externally networked source that the suspected pestware object originated from".

330. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

331. Thus, Kilpatrick in view of Piccard discloses limitation 1[e].

6. Limitation 1[f]

332. Limitation 1[f] of the '243 patent reads:

> *reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.*

333. Kilpatrick in view of Piccard discloses limitation 1[f] of the '243 patent.

334. Piccard discloses a "reporting module" that reports information about the identified malware and its source (e.g., "a Web site from which the file was downloaded") to a remote entity for further research:

> In particular, the Web site identification module 120 accesses the history log 116 to identify the Web address of the file. In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site. ***The reporting module 122 then reports this information to a remotely-located computer that is included in the computer network 104.*** This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the potential malware is, in fact, malware and whether the Web site is, in fact, a malware distribution site.
> (Ex. 1007, [0015], emphasis added).

335. A POSITA would have understood that the remotely-located computer as described in Piccard operates as an externally networked pestware research entity. This "remotely-located computer" serves the purpose of analyzing the information to determine whether the website is indeed a source of malware and subsequently reporting its findings. The "reporting module" is responsible for transmitting the necessary information to do such analysis.

> The third operation illustrated in FIG. 2 is to report that the Web site is a potential malware distribution site (block 204). In the illustrated embodiment, once the Web site identification module identifies the Web site, ***a reporting module (e.g., the reporting module 122) reports information regarding the Web site to a remotely-located computer that is connected to the protected computer.*** This information can identify the Web site as a potential malware distribution site, such as in terms of the domain name of the Web site. Alternatively, or in conjunction, this information can identify the file as including potential malware, such as in terms of the name of the file or the URL of the file. It is also contemplated that this information can include a representation of the file or can identify suspicious activities related to the file. ***This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the file does, in fact, include malware and whether the Web site is, in fact, a malware distribution site.*** If the Web site is determined to be a malware distribution site, a new or updated set of malware definitions can be generated based on content within the Web site, and the new or updated set of malware definitions can be provided to the

> protected computer. In addition, content within the Web site can be monitored on a periodic or other basis for new or updated malware. Also, a new or updated list of malware distribution sites can be generated so as to identify the Web site, and the new or updated list of malware distribution sites can be provided to the protected computer.
> (Ex. 1007, [0023], emphasis added).

336. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

337. Thus, Kilpatrick in view of Piccard discloses limitation 1[f].

338. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 1 obvious to a POSITA.

C. Claim 2 is rendered obvious by Kilpatrick in view of Piccard

339. Claim 2 of the '243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 2 reads,

> *The method of claim 1, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.*

340. As I explained with respect to limitation 1[e], Piccard discloses a "Web site identification module" that is invoked after a determination of malware by the "malware detection module", which accesses the history log to determine the external source of the object:

> In particular, when a file is accessed and downloaded, the Web browser 114 records a Web address of the file in the history log 116. A Web address typically specifies a location of a file within a Web site. For example, a Web address can be a Uniform Resource Identifier ("URI") of a file, ***such as a Uniform Resource Locator ("URL") of the file. It is also contemplated that a Web address can be defined in various other ways, such as using an Internet Protocol ("IP") address or any other identifier of a source of a file.***
> (Ex. 1007, [0013], emphasis added).
>
> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the

> file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

341. Accordingly, Piccard teaches identifying a website via a URL or an IP address of the file. In more detail, Piccard discloses that the "Web address of the file" can be a URL:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, ***the Web site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file,*** which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021], emphasis added).

342. Further, a POSITA would have understood that a web URL and an I.P. address are commonly convertible identifiers via any domain name system (DNS) server. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

343. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 2 obvious to a POSITA.

D. Claim 3 is rendered obvious by Kilpatrick in view of Piccard

344. Claim 3 of the ’243 patent, which depends from Claim 2, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 3 reads,

> *The method of claim 2, wherein the identifying includes accessing an activity log that includes information that relates the activity to the suspected pestware object.*

345. As I explained with respect to limitation 1[b], Kilpatrick collects system call information:

> As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210 based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).
>
> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. These wrapper instances wrap their processes by intercepting some or all of the system calls that are made by the process. This interception effectively puts the wrappers in complete control of their processes interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:37-55).
>
> The intrusion detection system therefore runs in two modes: record mode and detect mode. In record mode, the intrusion detection system generates a normative sequence database for each program executed. A per-program database for every program executed is thereby created.
> (Ex. 1008, 5:54-58).

346. Kilpatrick teaches monitoring individual programs whose identities are known to the wrapper doing the monitoring. A POSITA would have understood that the executable program

logs of Kilpatrick would relate the system call activity to programs in directories and that the system call logs of Kilpatrick would be combined with the logs of Piccard to relate file downloads with new or anomalous system calls. In addition, a POSITA would have understood that system call logs, such as those taught by Kilpatrick, identify the process that made the system call. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity (e.g., the claimed "suspected pestware object"), and a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs. Thus, a POSITA would have understood the system calls intercepted for a given program to be related to the activity of the object (e.g., file) identified.

347. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 3 obvious to a POSITA.

E. Claim 4 is rendered obvious by Kilpatrick in view of Piccard

348. Claim 4 of the '243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 4 reads,

> *The method of claim 1, wherein the recorded history resides in a browser cache, which includes information about files downloaded to the computer and a source of each of the files.*

349. As I explained with reference to limitation 1[e], Piccard, when a malware file is identified, uses its history log to identify the source of the file. Piccard describes the history log as follows:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module

> determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, the Web site identification module can access the Web browser's history log to ***identify a Web address of the file***. As described previously, the Web address of the file can be a URL of the file, which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
>
> As another example, the Web site identification module can generate the download history log based on the Web browser's history log. In this example, the Web site identification module can access the Web browser's history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0021]-[0022], emphasis added).

350. Piccard continues, emphasizing the performance benefits of having such a history log:

> By including such salient information in the download history log, the Web site identification module ***can accelerate and simplify a search process***. Further acceleration and simplification of the search process can be achieved by filtering out duplicative entries, such as in the event a same version of a file is downloaded multiple times from a Web site. It is also contemplated that the Web site identification module can generate the download history log independently of the Web browser's history log.
> (Ex. 1007, [0022], emphasis added).

351. The '243 patent specification describes "browser cache" as follows:

> The browser history 240 in this embodiment includes a listing of web sites and respective times that the web sites were visited by the user. The browser cache 242 in this embodiment includes files, cookies, and other objects cached in connection with use of a browser of the protected computer (e.g.,

> Internet Explorer or Mozilla (not shown)). As depicted, the browser cache 242 includes a cache index 243 that includes a listing, which associates the content of the browser cache 242 with URLs and time stamps. As discussed further herein, the ***cache index 243 provides an efficient means for identifying the times objects were retrieved and the URLs that the objects were retrieved from.***
> (Ex. 1001, 6:24-35, emphasis added).

352. A POSITA would have understood browser cache in the context of this claim to correspond to browser history log. Thus, Piccard's teachings about associating file names of downloaded files with URLs (e.g., a "Web address") align with the '243 patent's teachings about its browser cache, specifically the "cache index 243", including the efficiency of such an index.

353. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

354. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 4 obvious to a POSITA.

F. Claim 5 is rendered obvious by Kilpatrick in view of Piccard

355. Claim 5 of the '243 patent, which depends from Claim 1, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 5 reads,

> *The method of claim 1, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

356. As I explained in my analysis of Claim 3 (which I incorporate herein by reference), Kilpatrick discloses monitoring individual programs whose identities are known to the wrapper doing the monitoring and monitoring individual programs based on their directory (i.e., file) locations. (Ex. 1008, 4:37-62, 5:54-58). A POSITA would have understood that the executable

program logs of Kilpatrick would relate the system call activity to programs in directories and that the system call logs of Kilpatrick would be combined with the logs of Piccard to relate file downloads with new or anomalous system calls.

357. As I explained in my analysis of Claim 4 (which I incorporate herein by reference), Piccard discloses accessing "recorded history" that resides in a "browser cache" by extracting such information from the web browser's history log:

> As another example, the Web site identification module can ***generate the download history log based on the Web browser's history log***. In this example, the Web site identification module can access the Web browser's history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0022], emphasis added).

358. Furthermore, Piccard discloses accessing "recorded history" that resides in "a browser history" by accessing the browser's history log directly:

> In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. … Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021]).

359. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

360. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 5 obvious to a POSITA.

G. Claim 6 is rendered obvious by Kilpatrick in view of Piccard

361. Each of the limitations of Claim 6 of the '243 patent is rendered obvious by Kilpatrick in view of Piccard as described below.

1. Limitation 6[a]

362. Limitation 6[a] of the ’243 patent reads:

A system for identifying a source of activity on a computer that is indicative of pestware including:

363. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 6 of the ’243 patent is considered limiting, Kilpatrick in view of Piccard discloses the preamble of Claim 6.

364. Kilpatrick discloses a system of intrusion detection that monitors applications for anomalous behaviors:

> The present invention relates generally to intrusion detection, and more particularly, to systems and methods for implementing real-time sequence-based anomaly detection.
> (Ex. 1008, 1:9-11).

365. Kilpatrick explains the purpose of intrusion detection as follows:

> The second line of defense of the network security plan is intrusion detection. After an intruder has successfully penetrated the perimeter and gained access to systems within the protected network, a mechanism must exist to quickly detect the intruder and minimize the damage that the intruder can inflict.
> (Ex. 1008, 1:21-27).

> A real-time sequence-based anomaly detection system is disclosed. In a preferred embodiment, the intrusion detection system is incorporated as part of a software wrapper. Event abstraction in the software wrapper enables the intrusion detection system to apply generically across various computing platforms. Real-time anomaly detection is enabled through the definition of a distance matrix that defines allowable separation distances between pairs of system calls.
> (Ex. 1008, Abstract).

> In accordance with the present invention, a software wrapper provides a dynamically installable, non-bypassable, operating system extension that intercepts a selected program’s system calls. For each system call it intercepts, a wrapper may perform functions such as s (1) add functionality (intrusion detection, auditing, encryption) to the system call, (2) transform the system call (e.g. by replacing it with another), or (3) prevent the system call's execution (e.g., perform access control).

(Ex. 1008, 4:17-25).

366. Kilpatrick further discloses that its techniques operate on UNIX and can readily be extended to work in Windows:

> FIG. 2 illustrates the operation of a software wrapper 210. Although software wrapper 210 is described in the example context of a UNIX environment, the operation of software wrappers as described below can also be extended to other operating system environments (e.g., Windows NT™) as would be apparent.
> (Ex. 1008, 4:26-31).

367. Piccard teaches a method for identifying the source of malware by examining web traffic history:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

368. Piccard also discloses a computer system that implements its methods:

> FIG. 1 illustrates a computer system 100 that is implemented in accordance with an embodiment of the invention. The computer system 100 includes at least one protected computer 102, which is connected to a computer network 104 via any wire of wireless transmission channel. In general, the protected computer 102 can be a client computer, a server computer, or any other device with data processing capability. Thus, for example, the protected computer 102 can be a desktop computer, a laptop computer, a handheld computer, a tablet computer, a personal digital assistant, a cellular telephone, a firewall, or a Web server. In the illustrated embodiment, the protected computer 102 is a client computer and includes conventional client computer components, including a Central Processing Unit ("CPU") 108 that is connected to a network connection device 110 and a memory 112.
> (Ex. 1007, [0012]).

369. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. Kilpatrick is directed to the identification of processes exhibiting abnormal

behavior and anticipates the need for appropriate responses to such processes. Piccard teaches identifying the origin of pestware that has been detected by techniques like those taught in Kilpatrick. Kilpatrick's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

370. Thus, Kilpatrick in view of Piccard discloses limitation 6[a].

2. Limitation 6[b]

371. Limitation 6[b] of the '243 patent reads:

> *an activity monitor configured to monitor API calls and to store a history of at least a portion of the API calls in an activity log, wherein the activity monitor includes a kernel-mode driver adapted to intercept the API calls;*

372. Kilpatrick discloses limitation 6[b] of the '243 patent.

373. Kilpatrick claims "an event abstraction module that translates a stream of events into a generic event language format". (Ex. 1008, Claim 7). Kilpatrick further describes this module:

> In accordance with the present invention, software wrapper 210 monitors sequences of system calls that are passed to software wrapper 210. Software wrapper 210 enables a form of event abstraction through the use of characterized system call interfaces. Characterized system call interfaces allow software wrapper 210 to be portable between platforms when the APIs of the platforms can be described using the same attributes.
> (Ex. 1008, 4:63-5:3).

374. Kilpatrick further explains that this a "kernel-loadable module" monitors API calls made by an application:

> As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210 based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode

> and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).
>
> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. These wrapper instances wrap their processes by intercepting some or all of the system calls that are made by the process. This interception effectively puts the wrappers in complete control of their processes' interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:37-55).

375. Kilpatrick discloses storing the monitored API calls into a rolling "event window", which is later used for analysis:

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
> (Ex. 1008, 8:33-38).

376. Thus, Kilpatrick discloses limitation 6[b].

3. Limitation 6[c]

377. Limitation 6[c] of the '243 patent reads:

> *a heuristics module configured to identify an activity on the computer that is indicative of pestware residing on the computer and to analyze the activity to determine whether one or more weighted factors associated with the activity exceeds a threshold; and*

378. Kilpatrick discloses limitation 6[c] of the '243 patent.

379. Kilpatrick claims "a detection module that synchronously determines whether said stream of generic events includes an anomalous sequence of events". (Ex. 1008, Claim 7). Kilpatrick further describes this module:

> In a preferred embodiment, event abstraction through generic software wrappers is used to implement a multi-platform sequence-based anomaly detection wrapper. As noted, sequence-based anomaly detection is based upon the calculation of an anomaly value for a program's execution of an event sequence relative to a pre-computed database of event sequences.
> (Ex. 1008, 5:47-53).

380. The sequences are collected in "event window 300" and analyzed by comparing them with a distance matrix created from known-good training data:

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
>
> At this point, the system determines whether the new sequence of system calls in event window 300 is a recognized sequence. In accordance with the present invention, the determination process is not based upon a literal comparison of entire sequences of system calls. Rather, multiple pairs of system calls are compared relative to the predefined contents of distance matrix 220.
> (Ex. 1008, 8:33-45).

381. If such comparison reveals that the system call sequence exists in the distance matrix, no further processing is necessary:

> If each system call pair has been determined to exist in distance matrix 220, then the sequence of system calls that resides in event window 300 represents a known sequence of system calls. The process then loops back to step 502 where the next system call is received.
> (Ex. 1008, 9:3-8).

382. However, if the sequence was not found, Kilpatrick teaches performing anomaly analysis on the sequence by measuring the magnitude of the difference between known behavior (represented in the distance matrix) and the observed unknown/potentially abnormal behavior:

> If, at step 508, one of the system call pairs has been determined not to exist in distance matrix 220, then the system call sequence in event window 300 has not been seen before. In this case, the detect process proceeds to step 510, where anomaly analysis processing is performed***. In the anomaly analysis processing step, the magnitude of the anomaly is identified. This***

> ***magnitude measure represents the extent to which the system call sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220.*** In general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion.
>
> In a preferred embodiment of the present invention, a levenshtein distance is used as a measure of the magnitude of the difference between a system call sequence in a stored profile and the system call sequence that is generated by a program.
> (Ex. 1008, 9:8-23, emphasis added).

383. In other words, Kilpatrick discloses comparing a new system call sequence to other collected system calls in the distance matrix (Ex. 1008, 9:7-66) using a Levenshtein distance between each system call pair. These distances are then combined into a single magnitude of the difference which still "represents the extent to which the system call sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220." A POSITA would have understood that these distances are weighted to combine to a single magnitude of the difference. (*See* Ex. 1008, Equation 1, 9:60-65). Kilpatrick then discloses that once the "magnitude" or "distance" calculations are complete, the system can determine that the process is abnormal if the distance exceeds a threshold:

> The minimum value across all possible values of test head therefore represents the minimum levenshtein distance across all known sequences of system calls. Based upon the value of the minimum levenstein distance, the intrusion detection system can then determine whether an intrusion alert, or other forms of intrusion detection processing, should be initiated. For example, if the minimum levenstein distance is greater than a predefined threshold, the intrusion detection system can initiate steps to terminate the abnormal process.
> (Ex. 1008, 11:19-28).

384. Further, as described above, Kilpatrick also discloses that "in general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion", which a POSITA would have understood to weight variations

as an indicator of intrusion activity and determine a significant likelihood of intrusion. A POSITA would have understood comparing a number of distance values against a predefined threshold by using Kilpatrick's "minimum value" involves weighting the other values. (Ex. 1008, 11:19-28).

385. Thus, Kilpatrick discloses limitation 6[c].

4. Limitation 6[d]

386. Limitation 6[d] of the '243 patent reads:

> *a research portion configured to access, in response to a prompt from the heuristics module, a first set of recorded information on the computer that relates the activity to at least one file residing on the computer, and wherein the research portion is configured to access a second set of recorded information on the computer that relates the at least one file to an externally networked source from which the file was received; and*

387. Kilpatrick in view of Piccard discloses limitation 6[d] of the '243 patent.

388. Kilpatrick discloses collecting system call logs and sequences into a sliding window that is used to analyze the logs as described regarding limitation 6[b] (which I incorporate herein by reference). Kilpatrick describes using the collected system call events to determine whether there was an intrusion:

> If, at step 508, one of the system call pairs has been determined not to exist in distance matrix 220, then the system call sequence in event window 300 has not been seen before. In this case, the detect process proceeds to step 510, where anomaly analysis processing is performed***. In the anomaly analysis processing step, the magnitude of the anomaly is identified. This magnitude measure represents the extent to which the system call sequence in event window 300 differs from sequences that are represented by the data in distance matrix 220.*** In general, a small variation would indicate a lower likelihood of intrusion while a large variation would indicate a significant likelihood of intrusion.

> In a preferred embodiment of the present invention, a levenshtein distance is used as a measure of the magnitude of the difference between a system call sequence in a stored profile and the system call sequence that is generated by a program.
> (Ex. 1008, 9:8-23, emphasis added).

389. A POSITA would have understood that system call logs identify the process that made the system call. A POSITA would have understood that the executable program logs of Kilpatrick would relate the system call activity to programs in directories and that the system call logs of Kilpatrick would be combined with the logs of Piccard to relate file downloads with new or anomalous system calls. A POSITA would have known it to be possible, and would found it obvious and routine, to identify the file from which the process was launched. (*See* Section V.C. on known techniques for programmatically obtaining the file name of the executable based on the process ID). Such a file would then represent the source of the activity, as a POSITA would have understood process ID and file locations / paths to be common convertible identifiers for executable programs, and the information associating the system call activity to the file would represent the claimed "a first set of recorded information on the computer that relates the activity to at least one file residing on the computer".

390. Piccard teaches that its "Web site identification module" starts its search of the history log after determination by the "malware detection module" that a file includes potential malware:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, ***once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded.*** The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, the Web site identification module can access the Web browser's history log to

> identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file, which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021], emphasis added).

391. The purpose of Piccard's "Web site identification module" is to associate the suspicious file with its URL source. Thus, Piccard's "history log" teaches the claimed "a second set of recorded information on the computer that relates the at least one file to an externally networked source from which the file was received", which is accessed "in response to a prompt from the heuristics module" (e.g., the "malware detection module"). The malware detection module of Piccard analyzes files that have been downloaded and, in the combined Kilpatrick-Piccard system, analyzes files identified as the sources of anomalous system call sequences as described regarding claim limitations 6[b] and [c].

392. Thus, Kilpatrick in view of Piccard discloses limitation 6[d].

5. Limitation 6[e]

393. Limitation 6[e] of the '243 patent reads:

> *a reporting portion configured to generate a report that identifies the externally networked source of the file and to report an identity of the externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the externally networked source is a source of pestware.*

394. Kilpatrick in view of Piccard discloses limitation 6[e] of the '243 patent.

395. Piccard discloses a "reporting module" that reports information about the identified malware and its source (e.g., "a Web site from which the file was downloaded") to a remote entity for further research:

> In particular, the Web site identification module 120 accesses the history log 116 to identify the Web address of the file. In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site. ***The reporting module 122 then reports this information to a remotely-located computer that is included in the computer network 104.*** This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the potential malware is, in fact, malware and whether the Web site is, in fact, a malware distribution site.
> (Ex. 1007, [0015], emphasis added).

396. A POSITA would have understood the remotely-located computer as described in Piccard operates as an externally networked pestware research entity. This "remotely-located computer" serves the purpose of analyzing the information to determine whether the website is indeed a source of malware and subsequently reporting its findings. The "reporting module" is responsible for transmitting the necessary information to do such analysis.

> The third operation illustrated in FIG. 2 is to report that the Web site is a potential malware distribution site (block 204). In the illustrated embodiment, once the Web site identification module identifies the Web site, ***a reporting module (e.g., the reporting module 122) reports information regarding the Web site to a remotely-located computer that is connected to the protected computer.*** This information can identify the Web site as a potential malware distribution site, such as in terms of the domain name of the Web site. Alternatively, or in conjunction, this information can identify the file as including potential malware, such as in terms of the name of the file or the URL of the file. It is also contemplated that this information can include a representation of the file or can identify suspicious activities related to the file. ***This information as well as any additional relevant information can be analyzed at the remotely-located computer to determine whether the file does, in fact, include malware and whether the Web site is, in fact, a malware distribution site.*** If the Web site is determined to be a malware distribution site, a new or updated set of malware definitions can be generated based on content within the Web site, and the new or updated set of malware definitions can be provided to the

> protected computer. In addition, content within the Web site can be monitored on a periodic or other basis for new or updated malware. Also, a new or updated list of malware distribution sites can be generated so as to identify the Web site, and the new or updated list of malware distribution sites can be provided to the protected computer.
> (Ex. 1007, [0023], emphasis added).

397. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

398. Thus, Kilpatrick in view of Piccard discloses limitation 6[e].

H. Claim 7 is rendered obvious by Kilpatrick in view of Piccard

399. Claim 7 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 7 reads,

> *The system of claim 6, wherein activity monitor is configured to store an identity of each process that made each of the API calls in the stored history, and wherein the activity monitor is configured to store an identity of each file that corresponds to each the processes that made the API calls in the stored history so as to create a relation between each API call and at least one file.*

400. In the combined Kilpatrick-Piccard system, in which a POSITA is motivated to identify the source of the suspected malicious file, it would have been obvious to both obtain the identity of the file (e.g., through known techniques detailed in Section V.C.) and to store that identity for future reference, in order to gain processing efficiency by acquiring the information only once. Thereafter, since the stored activity includes the API call and the process identity associated with it, it would have been obvious to a POSITA that the identity of the file corresponding to the process can be trivially obtained (e.g., by storing it alongside the activity in the log, or by looking up the stored file name based on the process ID stored in the activity log).

401. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 7 obvious to a POSITA.

I. Claim 8 is rendered obvious by Kilpatrick in view of Piccard

402. Claim 8 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 8 reads,

> *The system of claim 6, wherein the heuristics module is configured to receive information about the API calls from the activity monitor and determine whether the API calls are indicative of pestware.*

403. Kilpatrick discloses wrapping processes to be monitored, such that their API calls (e.g., system calls) pass through the monitor first:

> As illustrated in FIG. 2, operating within the UNIX environment are programs 202A and 202B. Program 202B is an unwrapped program whose system calls pass directly to an internal application programming interface (API). Program 202A, on the other hand, is "wrapped' by software wrapper 210. As illustrated, the kernel-loadable module intercepts all system calls as they are made by a program 202A, and passes the system calls to wrapper 210 based on criteria (i.e., a wrapper specification (WS)) specified when wrapper 210 is loaded. In this framework, wrapper 210 runs in kernel mode and its execution environment is provided in the kernel loadable module. This execution environment is referred to as the wrapper enforcement layer (WEL).
>
> The WEL tracks running processes and evaluates activation criteria at appropriate times to activate new wrapper instances for processes. These wrapper instances wrap their processes by intercepting some or all of the system calls that are made by the process. This interception effectively puts the wrappers in complete control of their processes' interactions with the operating system and with other processes. For each system call, wrappers can observe and/or modify the parameters specified by the caller and the values returned by the operating system.
> (Ex. 1008, 4:33-55).

404. As I discussed in Section V.B., a POSITA would have understood that Windows system calls, such as those being recorded by a kernel-mode driver, relate to the claimed "API calls" of this limitation. (*See* Section V.B.). For example, an API call of CreateProcess indicates

one process (perhaps a malicious process) attempting to spawn another process, and monitoring calls to that API would allow analysis of such behavior:

> One way to accomplish that would be to establish a system-wide API interceptor that monitors calls made to the CreateProcess function (actually, to both the Ansi and Unicode versions of this function). Whenever an application issues a call to one of these functions in order to create a new process, your interceptor will gain control and perform whatever processing is necessary.
> (Ex. 1013, p. 1).

405. Kilpatrick discloses that the "detection process" (which, as discussed with reference to limitation 6[c], teaches the claimed "heuristics module") receives system call information from the wrapper:

> The detection process begins at step 502 when a new system call is received. At step 504, the new system call is inserted into event window 300. In this process, the remaining system calls in event window 300 are each shifted one position, with the oldest system call being dropped out of event window 300.
> (Ex. 1008, 8:33-38).

406. Thereafter, "the detect process proceeds to step 510, where anomaly analysis processing is performed", as I explained with reference to limitation 6[c]. (Ex. 1008, 9:11-13; *see also* my analysis of limitation 6[c]).

407. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 8 obvious to a POSITA.

J. Claim 9 is rendered obvious by Kilpatrick in view of Piccard

408. Claim 9 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 9 reads,

> *The system of claim 6, wherein the first set of recorded information and the second set of recorded information reside in the same log file on a file storage device of the computer.*

409. As I discussed in Section V.A., a POSITA would have understood the value of storing various sets of logging, auditing, or security information (e.g., the claimed "recorded information") within a common file or database on a computer, and it was commonly practiced in the art. (*See* Section V.A.; *see also* Ex. 1010, pp. 15-17).

410. As I discussed with respect to limitation 6[d], the "recorded information" stored and accessed by Kilpatrick was stored on the computer, and the "recorded information" stored and accessed by Piccard was also stored on the computer. A POSITA implementing the combination would have been motivated by the benefits enumerated by Rinnan to store these sets of information together, perhaps in a normalized form such as "a process that breaks an event into its component parts and places them into individual database fields." (Ex. 1010, p. 16).

411. A POSITA would have understood that a database, in order to preserve its information long-term, is stored in a file on a file storage device.

412. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 9 obvious to a POSITA.

K. Claim 10 is rendered obvious by Kilpatrick in view of Piccard

413. Claim 10 of the '243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 10 reads,

> *The system of claim 6, wherein the externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.*

414. As I explained with respect to limitation 6[d], Piccard discloses a "Web site identification module" that is invoked after a determination of malware by the "malware detection module", which accesses the history log to determine the external source of the object:

> In particular, when a file is accessed and downloaded, the Web browser 114 records a Web address of the file in the history log 116. A Web address typically specifies a location of a file within a Web site. For example, a Web address can be a Uniform Resource Identifier ("URI") of a file, ***such as a Uniform Resource Locator ("URL") of the file. It is also contemplated that a Web address can be defined in various other ways, such as using an Internet Protocol ("IP") address or any other identifier of a source of a file.***
> (Ex. 1007, [0013], emphasis added).
>
> In particular, once a file is downloaded using the Web browser 114, the malware detection module 118 analyzes the file to determine whether the file includes potential malware. ***If the file is determined to include potential malware, the Web site identification module 120 determines a Web address from which the file was received.*** In particular, the Web site identification module 120 ***accesses the history log 116 to identify the Web address of the file.*** In such manner, the Web site identification module 120 can identify a Web site from which the file was downloaded and, thus, can identify the Web site as a potential malware distribution site.
> (Ex. 1007, [0015]).

415. Accordingly, Piccard teaches identifying a website via a URL or an IP address of the file. In more detail, Piccard discloses that the "Web address of the file" can be a URL:

> The second operation illustrated in FIG. 2 is to search a download history log to identify a Web site from which the file was downloaded (block 202). In the illustrated embodiment, once the malware detection module determines that the file includes potential malware, a Web site identification module (e.g., the Web site identification module 120) accesses the download history log to identify the Web site from which the file was downloaded. The download history log serves to provide a record of downloading events. For example, the download history log can be a Web browser's history log (e.g., the history log 116). In this example, ***the Web site identification module can access the Web browser's history log to identify a Web address of the file. As described previously, the Web address of the file can be a URL of the file,*** which can have the following format: http://www.DomainName.com/Subdirectory/FileName.html, where "http://" specifies a communication protocol used to download the file, "www.DomainName" specifies a domain name of the Web site from which the file was downloaded, "/Subdirectory/" specifies a subdirectory within the Web site from which the file was downloaded, and "FileName.html" specifies a name of the file. Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.

(Ex. 1007, [0021], emphasis added).

416. Further, a POSITA would have understood that a web URL and an I.P. address are commonly convertible identifiers via any domain name system (DNS) server. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

417. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 10 obvious to a POSITA.

L. Claim 11 is rendered obvious by Kilpatrick in view of Piccard

418. Claim 11 of the ’243 patent, which depends from Claim 6, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 11 reads,

> *The system of claim 6 wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

419. As I explained in my analysis of Claim 10 (which I incorporate herein by reference), Piccard discloses accessing “recorded history” that resides in a “browser cache” by extracting such information from the web browser’s history log:

> As another example, the Web site identification module can ***generate the download history log based on the Web browser's history log***. In this example, the Web site identification module can access the Web browser's history log to extract salient information from the Web browser's history log, such as ***domain names of various Web sites and names of various files that were downloaded from the various Web sites***.
> (Ex. 1007, [0022], emphasis added).

420. Furthermore, Piccard discloses accessing “recorded history” that resides in “a browser history” by accessing the browser’s history log directly:

> In this example, the Web site identification module can access the Web browser's history log to identify a Web address of the file. … Thus, by searching the Web browser's history log using the name of the file, the Web site identification module can identify the Web site from which the file was downloaded, such as in terms of the domain name of the Web site.
> (Ex. 1007, [0021]).

421. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. I incorporate my analysis herein by reference.

422. Thus, I conclude that Kilpatrick in view of Piccard renders Claim 11 obvious to a POSITA.

M. Claim 12 is rendered obvious by Kilpatrick in view of Piccard

423. Each of the limitations of Claim 12 of the '243 patent is rendered obvious by Kilpatrick in view of Piccard as described below.

1. Limitation 12[a]

424. Limitation 12[a] of the '243 patent reads:

> *A non-transitory computer-readable medium including processor-executable instructions for identifying an origin of activity on a computer that is indicative of pestware, the instructions including instructions for:*

425. I understand that the preamble of a patent claim is not always considered limiting. However, to the extent that the preamble of Claim 12 of the '243 patent is considered limiting, Kilpatrick in view of Piccard discloses the preamble of Claim 12.

426. Kilpatrick discloses a system of intrusion detection that monitors applications for anomalous behaviors:

> The present invention relates generally to intrusion detection, and more particularly, to systems and methods for implementing real-time sequence-based anomaly detection.
> (Ex. 1008, 1:9-11).

427. Kilpatrick explains the purpose of intrusion detection as follows:

> The second line of defense of the network security plan is intrusion detection. After an intruder has successfully penetrated the perimeter and gained access to systems within the protected network, a mechanism must exist to quickly detect the intruder and minimize the damage that the intruder can inflict.
> (Ex. 1008, 1:21-27).
>
> A real-time sequence-based anomaly detection system is disclosed. In a preferred embodiment, the intrusion detection system is incorporated as part of a software wrapper. Event abstraction in the software wrapper enables the intrusion detection system to apply generically across various computing platforms. Real-time anomaly detection is enabled through the definition of a distance matrix that defines allowable separation distances between pairs of system calls.
> (Ex. 1008, Abstract).
>
> In accordance with the present invention, a software wrapper provides a dynamically installable, non-bypassable, operating system extension that intercepts a selected program's system calls. For each system call it intercepts, a wrapper may perform functions such as s (1) add functionality (intrusion detection, auditing, encryption) to the system call, (2) transform the system call (e.g. by replacing it with another), or (3) prevent the system call's execution (e.g., perform access control).
> (Ex. 1008, 4:17-25).

428. Kilpatrick further discloses that its techniques operate on UNIX and can readily be extended to work in Windows:

> FIG. 2 illustrates the operation of a software wrapper 210. Although software wrapper 210 is described in the example context of a UNIX environment, the operation of software wrappers as described below can also be extended to other operating system environments (e.g., Windows NT™) as would be apparent.
> (Ex. 1008, 4:26-31).

429. A POSITA would have understood that software in UNIX and Windows environments is typically stored on non-transitory computer-readable media such as hard drives or floppy disks, and that such software is executable by a processor on compatible machines. Designing a system with such storage would have been obvious, and a POSITA would have expected a high likelihood of success with such storage mechanisms.

430. Piccard teaches a method for identifying the source of malware by examining web traffic history:

> Embodiments of the invention include systems of managing malware. In one embodiment, a system includes a malware detection module configured to analyze a file of a protected computer to determine that the file is associated with malware. The system also includes a Web site identification module configured to search a download history log of the protected computer to identify a Web site from which the file was downloaded.
> (Ex. 1007, [0005]).

431. Piccard also discloses a computer system that implements its methods:

> FIG. 1 illustrates a computer system 100 that is implemented in accordance with an embodiment of the invention. The computer system 100 includes at least one protected computer 102, which is connected to a computer network 104 via any wire of wireless transmission channel. In general, the protected computer 102 can be a client computer, a server computer, or any other device with data processing capability. Thus, for example, the protected computer 102 can be a desktop computer, a laptop computer, a handheld computer, a tablet computer, a personal digital assistant, a cellular telephone, a firewall, or a Web server. In the illustrated embodiment, the protected computer 102 is a client computer and includes conventional client computer components, including a Central Processing Unit ("CPU") 108 that is connected to a network connection device 110 and a memory 112.
> (Ex. 1007, [0012]).
>
> In the illustrated embodiment, the memory 112 also stores a set of computer programs that implement the operations described herein. In particular, the memory 112 stores a malware detection module 118, a Web site identification module 120, a reporting module 122, and a malware removal module 124. As further described below, the various modules 118, 120, 122, and 124 operate to manage malware that can be present in the computer system 100.
> (Ex. 1007, [0014]).

432. As I discussed in Section X.A., a POSITA would have been motivated to combine Kilpatrick and Piccard. Kilpatrick is directed to the identification of processes exhibiting abnormal behavior and anticipates the need for appropriate responses to such processes. Piccard teaches identifying the origin of pestware that has been detected by techniques like those taught in

Kilpatrick. Kilpatrick's techniques are directed to the same purpose as Piccard's "malware detection module", and a POSITA would have expected a high likelihood of success in combining those teachings.

433. Thus, Kilpatrick in view of Piccard discloses limitation 12[a].

2. Limitation 12[b]-12[f]

434. The remaining limitations of Claim 12 are virtually identical to the corresponding limitations of Claim 1. I have summarized the differences in the following table:

1[b]: monitoring, using a kernel-mode driver, the computer for activity that is indicative of pestware, wherein the monitoring includes monitoring API calls and storing a history of at least a portion of the API calls in an activity log;	**12[b]:** monitoring, **with** a kernel-mode driver, the computer for activity that is indicative of pestware, wherein **the instructions for** monitoring include **instructions for** monitoring API calls and storing a history of at least a portion of the API calls in an activity log;
1[c]: analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;	**12[c]:** analyzing, heuristically, computer activity to determine whether one or more weighted factors associated with an activity exceeds a threshold so as to arrive at a determination that the activity is indicative of pestware;

1[d]: identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;	**12[d]:** identifying, based upon the activity, an object residing on the computer that is a suspected pestware object;
1[e]: accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and	**12[e]:** accessing, in response to the identifying an object, at least a portion of a recorded history of externally networked sources that the computer received files from so as to identify a reference to an identity of a particular externally networked source that the suspected pestware object originated from; and
1[f]: reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.	**12[f]:** reporting the identity of the particular externally networked source to an externally networked pestware research entity so as to enable the externally networked pestware research entity to research whether the particular externally networked source is a source of pestware.

435. I incorporate herein by reference my analyses of limitations 1[b]-1[f] and conclude that Kilpatrick in view of Piccard renders Claim 12 obvious to a POSITA.

N. Claim 13 is rendered obvious by Kilpatrick in view of Piccard

436. Claim 13 of the ’243 patent, which depends from Claim 12, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 13 reads,

> *The non-transitory computer-readable medium of claim 12, wherein the particular externally networked source is identified by an identifier selected from the group consisting of an I.P. address and a URL.*

437. This claim is virtually identical to Claim 2. I incorporate my analysis of Claim 2 herein by reference and conclude that Kilpatrick in view of Piccard renders Claim 13 obvious to a POSITA.

O. Claim 14 is rendered obvious by Kilpatrick in view of Piccard

438. Claim 14 of the ’243 patent, which depends from Claim 12, is rendered obvious to a POSITA by Kilpatrick in view of Piccard. Claim 14 reads,

> *The non-transitory computer-readable medium of claim 12, wherein the recorded history resides in at least one log selected from the group consisting of an activity log, a browser history, browser cache, browser settings, operating system settings, an event log, a debugging log, a firewall log, file information and monitoring software logs.*

439. This claim is virtually identical to Claim 5. I incorporate my analysis of Claim 5 herein by reference and conclude that Kilpatrick in view of Piccard renders Claim 14 obvious to a POSITA.

XI. THERE ARE NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

440. I am not aware of any objective indicia of non-obviousness related specifically to the ’243 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 ’243 patent.

XII. CONCLUSION

441. 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 September 7, 2023

By: _____
Seth James Nielson