

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.
Petitioner,

v.

WEBROOT, INC.
Patent Owner.

IPR2023-00655

U.S. Patent No. 8,418,250

DECLARATION OF SETH JAMES NIELSON

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	QUALIFICATIONS	2
III.	MATERIALS CONSIDERED	12
IV.	LEGAL STANDARDS	12
V.	Technical Background that Would Have Been Known to A POSITA	15
A.	VIRUSES AND BEHAVIOR-BASED VIRUS DETECTION.....	15
B.	JAVA’S MEDIATED SOFTWARE EXECUTION	18
VI.	OVERVIEW OF THE CHALLENGED PATENT	19
A.	THE ’250 PATENT SPECIFICATION AND CLAIMS.....	19

B.	PROSECUTION HISTORY	20
VII.	LEVEL OF SKILL IN THE ART	21
VIII.	Overview of the Prior art	21
A.	OVERVIEW OF US 2004/0111632 TO HALPERIN (“HALPERIN”).....	21
B.	OVERVIEW OF US 7,900,194 TO MANKINS	25
C.	OVERVIEW OF “ANOMALY INTRUSION DETECTION IN DYNAMIC EXECUTION ENVIRONMENTS” BY INOUE (“INOUE”).....	27
D.	OVERVIEW OF US 7,694,150 TO KIRBY (“KIRBY”).....	28
E.	OVERVIEW OF US 2004/0068652 TO CARPENTIER (“CARPENTIER”).....	30
IX.	GROUND 1: Claims 1-5, 12-17, and 24 are rendered obvious by HALPERIN IN VIEW OF Mankins	32
A.	Motivation to Combine Halperin and Mankins	32
B.	Claim 1 is rendered obvious by Halperin in view of Mankins	34
1.	Limitation 1[a]	35
2.	Limitation 1[b]	39
3.	Limitation 1[c]	42
4.	Limitation 1[d]	43
5.	Limitation 1[e]	44
6.	Limitation 1[f]	46
7.	Limitation 1[g]	51
8.	Limitation 1[h]	52
9.	Limitation 1[i]	53
10.	Limitation 1[j]	54
11.	Limitation 1[k]	55
C.	Claim 2 is rendered obvious by Halperin in view of Mankins	56

D.	Claim 3 is rendered obvious by Halperin in view of Mankins	58
E.	Claim 4 is rendered obvious by Halperin in view of Mankins	62
F.	Claim 5 is rendered obvious by Halperin in view of Mankins	64
G.	Claim 12 is rendered obvious by Halperin in view of Mankins	64
H.	Claim 13 is rendered obvious by Halperin in view of Mankins	70
1.	Limitation 13[a]	70
2.	Limitation 13[b]	71
3.	Limitation 13[c]	72
4.	Limitation 13[d]	72
5.	Limitation 13[e]	72
6.	Limitation 13[f]	73
I.	Claim 14 is rendered obvious by Halperin in view of Mankins	73
J.	Claim 15 is rendered obvious by Halperin in view of Mankins	74
K.	Claim 16 is rendered obvious by Halperin in view of Mankins	74
L.	Claim 17 is rendered obvious by Halperin in view of Mankins	74
M.	Claim 24 is rendered obvious by Halperin in view of Mankins	74
X.	GROUND 2: Claims 6 and 18 are rendered obvious by HALPERIN IN VIEW OF Mankins in further view of Arnold	75
A.	Motivation to Combine Halperin and Mankins and Arnold	75
B.	Claim 6 is rendered obvious by Halperin in view of Mankins in further view of Arnold	83
C.	Claim 18 is rendered obvious by Halperin in view of Mankins in further view of Arnold	87
XI.	GROUND 3: Claims 7-11, 19-23, and 25-30 are rendered obvious by HALPERIN IN VIEW OF Mankins in further view of Kirby in further View of Carpentier	88
A.	Motivation to Combine Halperin, Mankins, Kirby, and Carpentier	88

B.	Claim 7 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	94
C.	Claim 8 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	98
D.	Claim 9 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	99
E.	Claim 10 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	100
F.	Claim 11 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	102
G.	Claim 19 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	103
H.	Claim 20 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	103
I.	Claim 21 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	104
J.	Claim 22 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	104
K.	Claim 23 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	105
L.	Claim 25 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	105
1.	Limitation 25[a]	105
2.	Limitation 25[b]	105
3.	Limitation 25[c]	106
4.	Limitation 25[d]	106
5.	Limitation 25[e]	107
6.	Limitation 25[f]	107
7.	Limitation 25[f]	107
8.	Limitation 25[h]	107

9.	Limitation 25[i]	108
10.	Limitation 25[j]	108
M.	Claim 26 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	109
N.	Claim 27 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	109
O.	Claim 28 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	109
P.	Claim 29 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	110
Q.	Claim 30 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier	110
XII.	GROUND 4: Claims 1-5, 12-17, and 24 are rendered obvious by HALPERIN IN VIEW OF INOUE.....	111
A.	Motivation to Combine Halperin and Inoue	111
B.	Claim 1 is rendered obvious by Halperin in view of Inoue	113
1.	Limitation 1[a]	113
12.	Limitation 1[b]	117
13.	Limitation 1[c]	120
14.	Limitation 1[d]	121
15.	Limitation 1[e]	122
16.	Limitation 1[f]	124
17.	Limitation 1[g]	130
18.	Limitation 1[h]	131
19.	Limitation 1[i]	131
20.	Limitation 1[j]	132
21.	Limitation 1[k]	133
C.	Claim 2 is rendered obvious by Halperin in view of Inoue	135

D.	Claim 3 is rendered obvious by Halperin in view of Inoue	137
E.	Claim 4 is rendered obvious by Halperin in view of Inoue	139
F.	Claim 5 is rendered obvious by Halperin in view of Inoue	140
G.	Claim 12 is rendered obvious by Halperin in view of Inoue	141
H.	Claim 13 is rendered obvious by Halperin in view of Inoue	144
1.	Limitation 13[a]	144
7.	Limitation 13[b]	145
8.	Limitation 13[c]	146
9.	Limitation 13[d]	146
10.	Limitation 13[e]	146
11.	Limitation 13[f]	147
I.	Claim 14 is rendered obvious by Halperin in view of Inoue	147
J.	Claim 15 is rendered obvious by Halperin in view of Inoue	148
K.	Claim 16 is rendered obvious by Halperin in view of Inoue	148
L.	Claim 17 is rendered obvious by Halperin in view of Inoue	148
M.	Claim 24 is rendered obvious by Halperin in view of Inoue	149
XIII.	There are no objective indicia of nonobviousness.	149
XIV.	Conclusion.	149

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. 8,418,250 (“the ’250 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 ’250 patent are obvious in light of the prior art discussed below. The ’250 patent is directed to general application of known behavior analysis techniques to the identification of malware.

4. In summary, my conclusion is that the claims of the ’250 patent are obvious in light of the prior art discussed below. As set forth in detail in this declaration, I have reached the following conclusions:

- a. Claims 1-5, 12-17, and 24 are rendered obvious by Halperin in view of Mankins.
- b. Claims 6 and 18 are rendered obvious by Halperin in view of Mankins in further view of Arnold
- c. Claims 7-11, 19-23, and 25-30 are rendered obvious by Halperin in view of Mankins in further view of Kirby and Carpentier

- d. Claims 1-5, 12-17, and 24 are rendered obvious by Halperin in view of Inoue.

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 1011**. 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 '250 patent and its file history, the other documents and references as cited herein, and general technical references.

IV. LEGAL STANDARDS

50. 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.

51. I have been informed that in a parallel district court litigation the owner of the '250 patent is arguing that the '250 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.

52. I understand that a patent may include independent claims and dependent claims. An independent claim stands by itself and only includes the limitations it recites. A dependent claim can depend on (1) 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.

53. 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.

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

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

56. 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.

57. 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.

58. 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.

59. 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.

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

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

61. In this section of my Declaration, I discuss some of the background and context for the alleged invention in the ’250 patent that would have been known to a POSITA.

A. VIRUSES AND BEHAVIOR-BASED VIRUS DETECTION

62. Computer security is very hard to predict. It is hard to know how the attackers will attack us. Just as a simple example, the security community, as a whole at least, did not predict the ransomware tsunami that has impacted numerous personal and business systems around the globe and is now crippling cities. In retrospect, it’s somewhat obvious now that the conditions were ripe for such a catastrophe. Over the past few years we have also developed some very effective tools against such abominations, but we didn’t develop those defenses in advance because we didn’t predict the need.

63. One very striking exception to this lack of prescience is viruses, one of the earliest forms of malware. In 1987, years before the first true viruses appeared, Fred Cohen published his doctoral dissertation about computer viruses. (See Ex. 1009). He and several others described

how viruses would work, how they could spread, and accurately predicted that viruses would become a serious threat. See Computer Viruses Theory and Experiments by Fred Cohen.

64. But even more importantly, Dr. Cohen also proved in his dissertation that it would be impossible to write a computer program that could recognize all possible viruses. An important theoretical principle, foundational to all of computer science, establishes that a computer program cannot determine certain behaviors and attributes for all other possible computer programs. Dr. Cohen used this principle to prove that it is impossible to write a program that could examine all other programs and identify which ones were safe and which ones were dangerous. (Id).

65. Despite three decades of research, we still take use the same general anti-malware approaches that were developed in the 1990's. These approaches can be categorized by the generic approach or approaches they take to get around the limitations identified by Dr. Cohen. Some of the most common approaches are:

a. Malware Detection (As Many as Possible)

1.Static Analysis

1. Signatures

2. Heuristics

2.Dynamic Analysis

1. Behavior Blockers

2. Execution analyzers

b. System Hardening

1.Integrity checks and authenticated controls

2.“Safe” computer languages

66. At least to date, none of these approaches, either alone or collectively, have been able to stop all possible malware, which is why computer security companies are still in business. The goal of each approach is to cast the widest possible net in identifying and stopping malware. This means defending against the vast majority of all current threats and prevent the worst and most costly damage. Each method pursues the same goal from a different perspective.

67. Signature-based approaches are systems that identify known viruses and malware based on identifying sequences of data in the software. But, because signature-based scanners cannot find previously unseen malware, computers protected by the most up-to-date signatures database are, by definition, not protected against the next virus. Thus, antivirus researchers have been searching for truly generic antivirus detection from the beginning.

68. A “heuristic,” in computer science at least, is an “intelligent guess” that enables decision making when perfect information is not available. By 1995, antivirus researchers were pursuing heuristics evaluations to generically detect viruses (e.g., without signatures). Heuristics can be based either on the software instructions themselves or on their behavior while running.

69. Execution analyzers are an example of analyzing software while running. These kinds of tools attempt to execute as much of the target program as possible in an effort to understand what it does. This is sometimes called “detonation.” As with real world explosives that are intentionally set off in safe spaces by bomb squads, the target programs also have to be executed in a “safe space.” Accordingly, the program is executed where it can do no real damage, typically in an emulator or a virtual machine.

70. The definition of “virtual machine” is somewhat context-dependent, but tends to imply a fully simulated set of computer hardware. For some virtual machines, the simulation is so complete that real operating systems can be, and typically are required to be, installed on the virtual

hardware. Running a virus in a virtual machine is a generally safe way to try and see if the unknown software does harmful operations. This is sometimes called “sandbox analysis.”

B. JAVA’S MEDIATED SOFTWARE EXECUTION

71. As the Internet exploded in the mid 90’s, researchers investigated various “safe” computer languages. In theory, these languages would allow a user to run code from somewhere else (on the Internet) without worry that the code could do damage. The basic idea of such a language is that functions that can do damaging things are either disallowed or carefully monitored. One of the best known examples is the Java programming language.

72. Java was designed from the ground up for this purpose. Programs written in Java had no direct access to system resources such as file read and file write. Instead, Java programs used mediated functions that have a special virtual machine for checking security constraints first, then permitted the access if the security checks passed. Basically, a “safe” function guards access to an “unsafe” function, as demonstrated below:

```
public boolean mkdir(String path) throws IOException {
    SecurityManager security = System.getSecurityManager();
    if (security != null) {
        security.checkWrite(path);
    }
    return mkdir0();
}
```

Figure 1: Example of security check. This example shows the basic metaphor: the public method `mkdir` checks the system `SecurityManager` (which will throw an exception if the check does not pass) and then calls the low level private method `mkdir0`.

(See, e.g., Ex. 1010)

73. In the computer science world, we would say that the function “`mkdir()`” wraps the function “`mkdir0()`.” The general concept of one function wrapping another is as old as computer programming itself. It also follows a colloquial saying that computer scientists say only half-

jokingly: “all problems in computer science can be solved by another level of indirection.”
(Attributed to David Wheeler).

74. Java’s security mediation is not static. Rather, Java has security policies that can be configured for a virtual machine. These policies indicate which operations are allowed and under what conditions. This means that security can be customized for specific needs and circumstances.

VI. OVERVIEW OF THE CHALLENGED PATENT

A. THE ’250 PATENT SPECIFICATION AND CLAIMS

75. The ’250 patent relates to classifying, “objects” such as software, as malware. The patent describes a base computer that receives data about an object from remote computers. That is, the same object (or a similar object) is on the remote computers and they collect their information about the object at the base computer. The base computer uses this combined information to classify the object as malware or not. In the words of the ’250 patent:

a method of classifying a computer object as malware includes receiving at a base computer data about a computer object from each of plural remote computers on which the object or similar objects are stored. The data about the computer object received from the plural computers is compared in the base computer. The computer object is classified as malware on the basis of said comparison.
(Ex. 1001 at Abstract)

76. The patent is not particularly detailed about how the malware will be identified, but it states that the classification will be used using one or more characteristics such as, executable instructions, the size of the object, the name of the object, the storage location of the object, and so forth. (*Id.*)

B. PROSECUTION HISTORY

77. I understand that the '250 patent was issued based on patent application 11/477,807 which was filed on June 30, 2015. (See Ex. 1007, 821–884.)

78. At one point during the prosecution of the '250 patent, I understand that all pending claims (1-11, 13-23, 25-30, and 48-51) were rejected. Specifically, claims 1-10, 13-22, 25-28 and 30 were rejected under 35 U.S.C. § 103(a) as unpatentable over U.S. Patent Publication 2003/0177394 to Dozortsev in view of U.S. Patent Publication 2002/0099952 to Lambert. (See Ex. 1007, 307–318.) Also, claims 11, 23 and 29 were rejected under 35 U.S.C. § 103(a) as unpatentable over Dozortsev and Lambert in view of U.S. Patent Publication 2003/0084323 to Gales.

79. In an amendment filed June 6, 2015, Applicants responded to this rejection by amending the independent claims. Specifically, the Applicants modified the independent claims to require “... at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers; ... automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data.” (See Ex. 1007, 295–300, underlines showing changes to the claims).

80. I understand that the “automatic” generation of the mask was stressed as particularly important. In distinguishing the prior art, Applicants stated, “For example, as discussed at paragraphs 47 and 49 of Dozortsev, the objects are analysed centrally by a human analyst by [manually] running the file in a ‘sandbox’ environment or by examining the code...” (Ex. 1007, p. 467). As opposed to the human-driven analysis, Applicants emphasized their view

that “a human decision is not necessary to generate the mask or monitor operation.” (Ex. 1007, p. 143).

VII. LEVEL OF SKILL IN THE ART

81. Based on my knowledge and experience, it is my opinion that a person of ordinary skill in the art at the time of the earliest claimed priority date for the '250 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 network forensics. More education can supplement practical experience and vice versa.

82. I meet that definition of a person of ordinary skill in the art, and I did so on in and before June 2005. I also have worked with persons of ordinary skill in the art as part of my professional and academic experience, and I have an understanding of their skill level in and before June 2005.

VIII. OVERVIEW OF THE PRIOR ART

A. OVERVIEW OF US 2004/0111632 TO HALPERIN (“HALPERIN”)

83. US Patent Application Publication 2004/0111632 to Halperin (“Halperin”) is entitled, “system and method of virus containment in computer networks.” Halperin describes cooperative computers sharing suspicious events detected locally on each device with a central rendezvous point such as a server. Aggregating the data from each individual computer is used to detect malware and raise the appropriate alarms. (Ex. 1003 at Abstract). In more detail, Halperin goes on to explain how the system works to identify malicious code (malware).

The proposed system works in two phases to *identify malicious code* (from here on referred to as “viruses”). *The two phases are: learning and detecting.* In the learning phase all the sensors are fully operational in the same network that is to be protected by the virus detection system. Alternatively it is possible, but less preferable, to do the learning phases and detecting phases on different computer

networks. In such a case some kind of normalization processes may be required in porting the data from one network to another. In the learning phase all events are logged in a learning database similar to the database described above. This is done for a substantial period of time, for example, one week. In addition, the system gets retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system. Generally, commercial antivirus software can quite effectively identify an existence of a virus in the system after it has received the relevant updates from the antivirus vendor, usually within a few days of the initial virus outbreak. This input is logged in a virus outbreak database. The time period in which it is estimated that the virus was active and the computers that were affected are logged in the virus outbreak database. ***In some cases the existence of a virus is identified only a few days after the virus started its activity.*** For that reason, the learning database is analyzed and put into use only after a waiting period (for example, two weeks). It is assumed that there is a high probability that most of the viruses in the system will be identified within the waiting period. (Ex. 1003 at [0087], emphasis added).

84. In Figure 2, Halperin discloses containing a virus once identified as shown below.

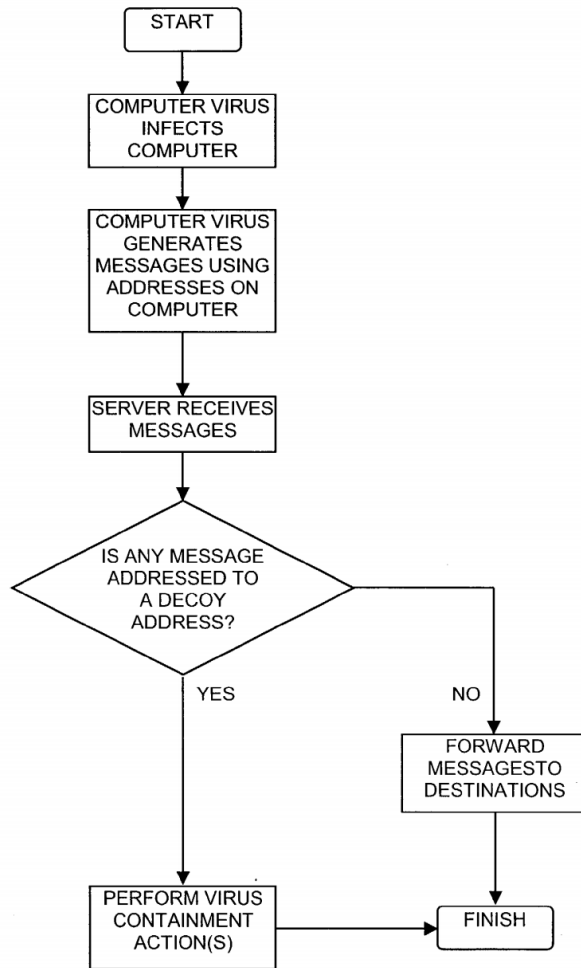


Fig. 2

85. Halperin also disclosed detecting a virus and transmitting information about the virus as shown in Figure 7, reproduced hereinbelow.

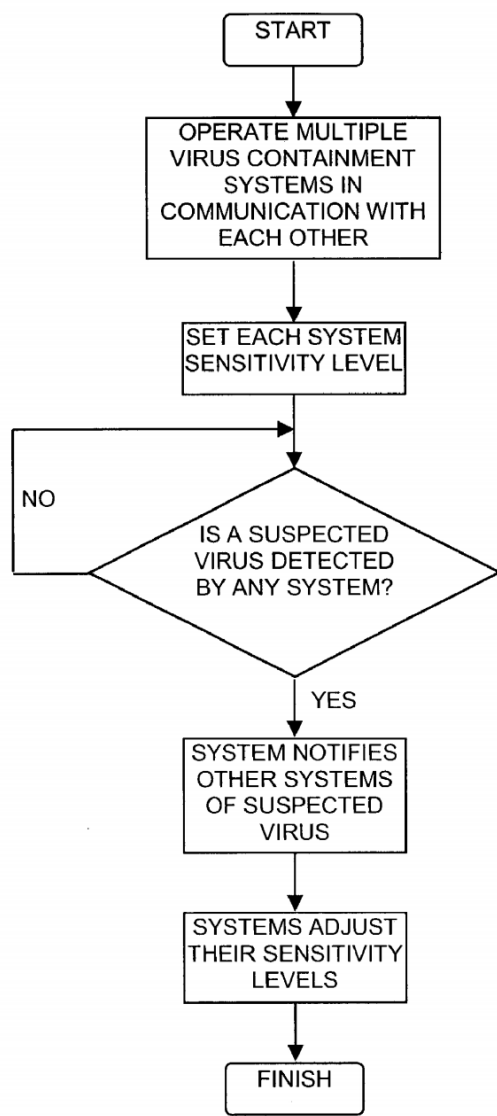


Fig. 7

86. Halperin also identifies viruses using behavior patterns as shown in Figure 11, reproduced hereinbelow.

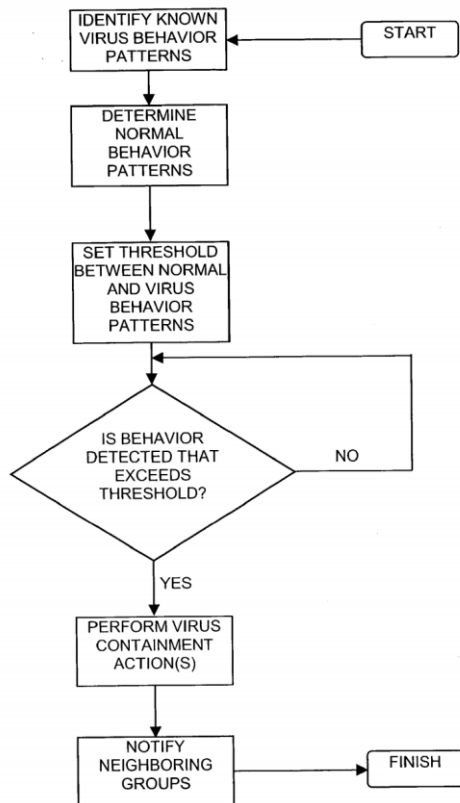


Fig. 11

87. A POSITA would have understood that a virus is malware.

B. OVERVIEW OF US 7,900,194 TO MANKINS

88. US Patent 7,900,194 to Mankins (“Mankins”) is entitled, “Kernel-based intrusion detection using bloom filters.” Mankins is directed toward creating a “data object” that “contains data representative of expected system-call behavior associated with a computer program.” During the execution of the software, “system-call behavior is compared with the data in the... data object.” (Ex. 1004 at Abstract).

89. In more detail, Mankins discusses using a “bloom filter” as the data object containing the expected behavior. As Mankins explains, the bloom filter can be designed to have a low false-positive rate. “Preferably, high-quality hash functions are used in the Bloom filter 175

to help guarantee that output is equally distributed over all possible values in the bit vector, which helps to minimize the occurrence of false positives.” (Ex. 1004 at 8:38-41).

90. The data stored in the bloom filter is derived from a system-call monitoring system. “In preferred embodiments, the monitoring subsystem 150 is configured to monitor system calls occurring in the OS kernel 120. The system calls can be associated with their initiating computer program(s) 130. A rolling log of system calls may be maintained.” (Ex. 1004 at 6:7-11). It is later stored in the bloom filter during the “build” process of the software. “The build subsystem 160 is configured to store the generated profiles of normal program behavior (e.g., sets of unique system-call sequences) compactly by using the Bloom filter 175.” (Ex. 1004 at 7:22-25).

91. In terms of operations, Mankins discloses learning a piece of software’s “normal” operations (e.g., system calls to the OS) automatically during a training phase. During the training session, the calls to the operating system are monitored and recorded as discussed above. The monitored data is encoded into a bloom filter that is associated with the software object (e.g., appended to it). When the program is running, its behavior is compared to the bloom filter. If the program attempts to deviate from the “trained” behavior, the system raises an alert.

To define normal behavior, a particular computer program 130 may be invoked and operated ***under training conditions*** in order to determine a set of operations initiated by the computer program 160. (Ex. 1004 at 6:63-66, emphasis added).

The build subsystem 160 is configured to ***store*** the generated profiles of normal program behavior (e.g., sets of unique system-call sequences) compactly by using ***[a] Bloom filter*** 175. (Ex. 1004 at 7:22-25, emphasis added).

The build subsystem 160 may be configured to associate a generated Bloom filter data object 180 to its corresponding computer program 130. In particular, the build subsystem 160 may cause the Bloom filter data object 180 to be ***appended to the executable image*** of the computer program 130 in a manner that will cause the Bloom filter data object 180 to be

automatically embedded in the OS kernel 120 upon invocation of the computer program 130.
(Ex. 1004 at 7:59-66, emphasis added).

The intrusion detection system 100 may be configured to ***automatically*** generate Bloom filter data objects 180 for computer programs 130 and then ***automatically*** use the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time.
(Ex. 1004 at 8:8-13, emphasis added)

92. Thus, Mankins bloom filter is attached to the software and subsequently indicates that kind of operations are considered “normal” and which are considered “abnormal.” As stated in the specification:

If an object is allowed to run on a remote computer 2... then operation of that object may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on. Thus, the mask can give an overall picture of the expected “normal behaviour” of that object.
(Ex. 1001 at 14:58-15:4).

C. OVERVIEW OF “ANOMALY INTRUSION DETECTION IN DYNAMIC EXECUTION ENVIRONMENTS” BY INOUE (“INOUE”)

93. The research paper “anomaly intrusion detection in dynamic execution environments” by Inoue, et al. (“Inoue”) discloses analyzing software for signs of malware. The software is run in a Java Virtual Machine (JVM) as part of a “training” session. During the training session, the calls to the operating system are detected as “profiling information.” The collected profile data is encoded into a special Java sandbox that the program must run in thereafter. If the program attempts to deviate from the “trained” behavior, it is stopped.

In the first, ***a sandbox profile is constructed*** by running the program with an instrumented JVM. ***During this training session, profiling information is recorded to the sandbox profile.*** The sandbox is initially empty and

grows during the training run by accumulating records for each unique behavior. Because nothing is added that isn't observed, each sandbox is customized to a given program and context in which it is executed. ***During sandbox execution, behavior that isn't in the profile is considered anomalous.***

(Ex. 1005 at p. 3, emphasis added)

94. Thus, Inoue's sandbox maps precisely with the concept of the mask as explained in the '250 patent. As stated in the specification:

If an object is allowed to run on a remote computer 2... then operation of that object may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on. Thus, the mask can give an overall picture of the expected "normal behaviour" of that object.

(Ex. 1001 at 14:58-15:4).

D. OVERVIEW OF US 7,694,150 TO KIRBY ("KIRBY")

95. US Patent 7,694,150 to Kirby ("Kirby") is entitled, "System and methods for integration of behavioral and signature based security." Kirby is directed towards identifying malware. For example, "[t]he behavioral monitoring correlates the actions performed to determine an undesirable object. A signature generator computes a realtime signature on the suspect object." (Ex. 1006 at Abstract). In more detail, Kirby teaches using behavioral identification of previously unseen malware and then the generation of an identifier. "Therefore, the behavioral approach analyzes the actions performed and the provenance of the data used, and determines whether an attempted activity is indicative of suspect activity. Further, one likely outcome of detecting and possibly preventing such suspect activity is the identification of an object or entity, such as a file, that is very likely the method of conveyance of the malicious code or instructions, or 'malware' to those of skill in the art." (Ex. 1006 at 2:21-42).

96. With respect to generating an identifier of the malware, Kirby discloses using hashing:

A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator accumulates successive realtime signatures in this manner for comparison with subsequent incoming transmissions.
(Ex. 1006 at 4:10-16).

97. The hashes, Kirby discloses, build up a set of signatures for subsequent comparison.
“A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator therefore accumulates a baseline of realtime signatures for comparison with subsequent incoming transmissions.” (Ex. 1006 at 7:41-62).

98. Kirby describes how the hash-based signature generated may be incorporated into a system.

FIG. 1 is a context diagram of an information processing environment 100 suitable for use with configurations of the invention. Referring to FIG. 1, the environment 100 includes one or more security agents 110 on a host computer 120. The host computer 120 couples to a public access network such as the Internet 130 via a local interconnection 140 such as a local area network (LAN) and a local physical or logical access point (e.g. firewall) 142 or other Internet connection or gateway. The LAN 140 also couples an AV server 150, a mail server 160, and a security master console (MC) 170. Further, the master console 170 couples to a signature repository 172 for manipulating signature and hash results, discussed further below.
(Ex. 1006 at 8:13-25).

99. Fig. 1, cited above, is reproduced herein below:

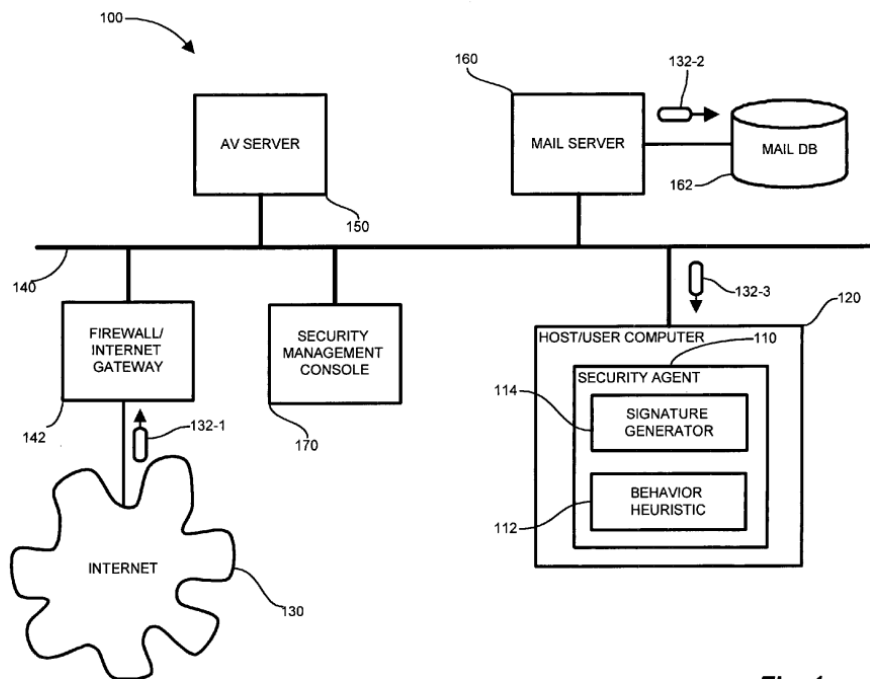


Fig. 1

100. Kirby explains further that “[t]he security agent 110 also includes a signature generator 114, such as a hash engine, for producing a signature or fingerprint corresponding to an object indicative of an undesirable transmission, such as a file containing a virus.” (Ex. 1006 at 8:43-46).

E. OVERVIEW OF US 2004/0068652 TO CARPENTIER (“CARPENTIER”)

101. US Published Patent Application 2004/0068652 to Carpentier, et. al (“Carpentier”), is entitled “Access to Content Addressable Data Over a Network.” Carpentier uses this same language to explain that the its disclosures are directed to “access to content addressable data over a network.” (Ex. 1013 at [0003]).

102. In Carpentier’s teachings, a “...cryptographic hash function used to generate the asset content identifier can be applied to the contents of the asset (or to information which is believed to be the original asset) at any time and the match between the asset and the asset content

identifier can then be verified.” (Ex. 1013 at [0034]). The content identifier is used to identify rather than “file name, date, location, or any other such label which may be changed independently of the asset contents.” (*Id.*). Using a content identifier, rather than an arbitrary label, “an asset may be obtained from any location and the integrity of the contents may be reliably verified by recomputing the asset hash and comparing it to a stored asset content identifier...” (Ex. 1013 at [0013]).

103. In addition to a hash over the contents of the file itself, Carpentier also discloses using a cryptographic hash on metadata about the file as stored in a “descriptor file.”

An example of a descriptor file is shown in FIG. 5. The other asset information included with the descriptor file may include directory information about where the files are located, file names, file sizes, time and date stamps for each file included, ownership of such files, and other meta data, as is described below. The descriptor file may contain similar descriptions of binary assets which were not originally files and which may not be used to create unique files. Such assets represent binary sequences which are records, elements, or clips of larger sequences or collections. These assets are treated in a manner similar to files and are returned to their user for processing through appropriately designed binary interfaces. (Ex. 1013 at [0031]).

104. Fig. 5, referenced above, is reproduced hereinbelow:

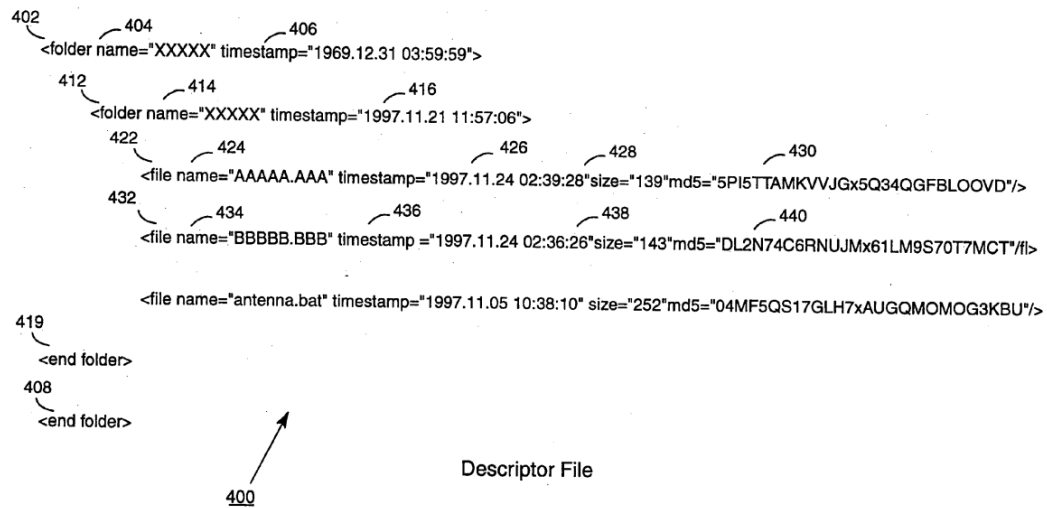


FIG. 5

105. Carpentier explains that the descriptor file is also hashed: “[a] cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed.” (Ex. 1013 at [0033]). The descriptor file is useful in identifying multiple content files (e.g., in a list) and the hash representing a pointer to the set of files. “Once the asset content identifier for each of the selected assets is generated, it is included in an asset list along with other relevant information in a descriptor file.” (Ex. 1013 at [0031]). “A cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed which, in one embodiment, becomes the e-CLIP by which the collected files may be referenced, found, and verified.” (Ex. 1013 at [0033]).

IX. GROUND 1: CLAIMS 1-5, 12-17, AND 24 ARE RENDERED OBVIOUS BY HALPERIN IN VIEW OF MANKINS

106. As explained in the sections and paragraphs below, Halperin in view of Mankins renders obvious the identified claims.

A. Motivation to Combine Halperin and Mankins

107. A POSITA would have been motivated to combine Halperin and Mankins. Both Halperin and Mankins are both directed to the same problem. Specifically, both references attempt

to protect computers on a network from intrusion. Halperin's approach is reactive insofar as it detects bad behavior and then acts to "contain" the infection. Mankins' approach is proactive insofar as it trains a guard system for acceptable behavior for a piece of software. The guard system then alerts system administrator if the software does not stick to the trained behavior.

108. Because of these synergies, a POSITA would have recognized the compatibility of these approaches. In particular, malware detection often includes both proactive and reactive systems. Because no one approach solves all malware problems, it is common to combine approaches to better obtain "defense in depth."

109. It would have been obvious to the POSITA that Mankins supports Halperin's goals. For example, Halperin discloses a system that learns as it goes, collecting data about "normal" and "abnormal" operations. Halperin, for example, discloses using increasing suspicious activity to eventually activate containment protocols.

In an alternative implementation of the buffer described above that is designed to reduce false alarms, should the server receive an invalid decoy message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified. During the increased delay period, ***should additional suspicious messages be received, or should other suspicious behavior be detected, if the user and/or system administrator who is authorized to do so has not indicated that the activity is not virus related, only then does the server perform one or more virus containment actions.*** If, however, during the increased delay period no other suspicious activity is detected, or if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.

(Ex. 1003 at [0059], emphasis added.)

110. Nevertheless, Halperin does identify that one of the problems of this approach is that there must be a trade off between false alarms (detecting a "good" program as a bad one) and not detecting malware at all. As Halperin explains, "... the [probability] result is compared to a

predefined threshold. If the calculated value is lower than the predefined threshold, the system can activate a virus alarm and possibly initiate virus containment actions. *The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.*” (Ex. 1003 at [0090], emphasis added).

111. Mankins offers another alternative to a simple false-alarm/less-detection trade-off. A program believed to be benign can have what is called a “bloom filter” appended to the program. The bloom filter identifies the program’s normal behavior. That way, if the software deviates from the normal behavior an alert is raised. As Mankins explains, the bloom filter can be designed to have a low false-positive rate.. “Preferably, high-quality hash functions are used in the Bloom filter 175 to help guarantee that output is equally distributed over all possible values in the bit vector, which helps to minimize the occurrence of false positives.” (Ex. 1004 at 8:38-41).

112. In other words, Mankins can be used as an active quarantine system. Because Halperin recognizes that suspicious activity can be a “false positive,” it would have been obvious to wrap the suspect software into a system that can continue to monitor it along with a guard of acceptable behavior.

113. A POSITA would have been motivated to add these protections to Halperin, preventing damage when malware is not detected and improving the false positive rates. Moreover, because of the ease of compatibility, the overlapping domains, and the use of standard functionalities, a POSITA would have had a strong expectation of success in combining these two references.

B. Claim 1 is rendered obvious by Halperin in view of Mankins

114. Each of the limitations of Claim 1 of the ’250 patent is rendered obvious by Halperin in view of Mankins as described below.

1. Limitation 1[a]

115. Limitation 1[a] of the '250 patent reads:

A method of classifying a computer object as malware, the method comprising:

116. 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 '250 patent is considered limiting, Halperin discloses a method of classifying a computer object as malware.

117. For example, Halperin discloses, “[a] method for detecting malicious activity in a computer network...” (Ex. 1003 at Abstract). Halperin goes on to explain that it detects malicious activity caused by viruses. In Figure 2, Halperin discloses containing a virus as shown below.

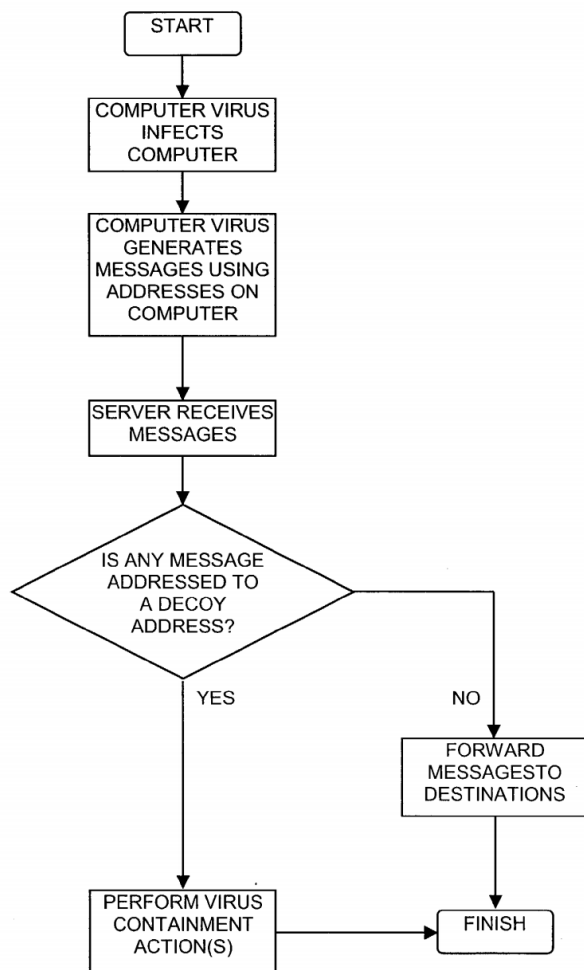


Fig. 2

118. Halperin also disclosed detecting a virus and transmitting information about the virus as shown in Figure 7, reproduced hereinbelow.

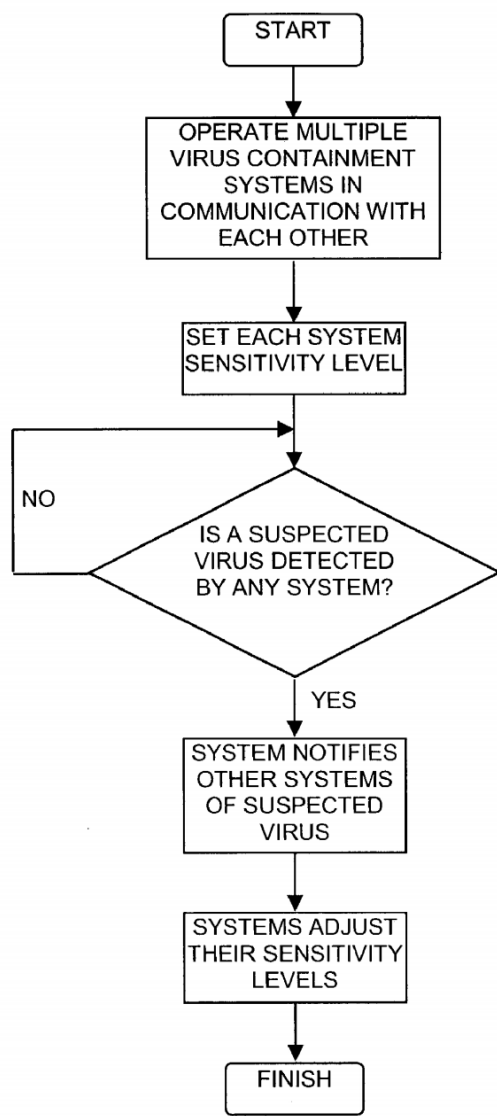


Fig. 7

119. Halperin also identifies viruses using behavior patterns as shown in Figure 11, reproduced hereinbelow.

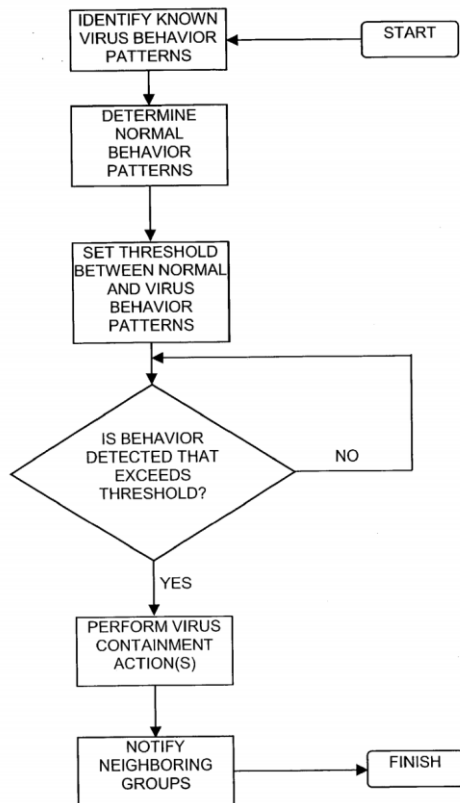


Fig. 11

120. A POSITA would have known that a virus is an example of malware.

121. Halperin also explains, “[t]he present invention relates to computer and computer network security in general, and more particularly to detection and prevention of malicious computer programs.” (Ex. 1003 at [0002]). A POSITA would have known that “malicious computer programs” are malware.

122. In more detail, Halperin goes on to explain how the system works to identify malicious code (malware).

The proposed system works in two phases to *identify malicious code* (from here on referred to as “viruses’). **The two phases are: learning and detecting.** In the learning phase all the sensors are fully operational in the same network that is to be protected by the virus detection system. Alternatively it is possible, but less preferable, to do the learning phases and detecting phases on different computer networks. In such a case some kind of normalization processes may be required in porting the data from one network to another. In the learning phase all events are logged in a learning database similar to the database described above. This is done

for a substantial period of time, for example, one week. In addition, the system gets retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system. Generally, commercial antivirus software can quite effectively identify an existence of a virus in the system after it has received the relevant updates from the antivirus vendor, usually within a few days of the initial virus outbreak. This input is logged in a virus outbreak database. The time period in which it is estimated that the virus was active and the computers that were affected are logged in the virus outbreak database. ***In some cases the existence of a virus is identified only a few days after the virus started its activity.*** For that reason, the learning database is analyzed and put into use only after a waiting period (for example, two weeks). It is assumed that there is a high probability that most of the viruses in the system will be identified within the waiting period. (Ex. 1003 at [0087], emphasis added).

123. Thus, I conclude that Halperin discloses limitation 1[a].

2. Limitation 1[b]

124. Limitation 1[b] of the '250 patent reads:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

125. Halperin discloses this limitation as broken down in the paragraphs below.

At a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored.

126. In order to identify malware, Halperin discloses a centralized server that receives data about a virus from multiple computers. Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.***

(Ex. 1003 at [0047], emphasis added).

127. Notably, this disclosure from Halperin expresses that the same behavior must be detected at two or more computers in order for the server to determine that a virus may have infected the machines. Thus, the data received is data about the same or similar virus (or “object”) on two different computers.

128. Although the data received by the central server is behavioral, Halperin is clear that specific virus software (and not just general unwanted behavior) can be identified. One approach disclosed by Halperin is the use of “decoy” or “bait” elements on the system that, when accessed, trigger an alert about the accessing software. As Halperin explains, “In the method of FIG. 7 a number of virus detection and containment systems are implemented... each system may use ***different system decoys*** and/or monitor different correlation parameters... if one system ***detects a suspected virus using a specific decoy*** or correlation parameter, other systems may heighten their sensitivity level related to that decoy or correlation parameter.” (Ex. 1003 at [0061]; emphasis added). Figure 7 of Halperin, which illustrates part of this process, is reproduced hereinbelow:

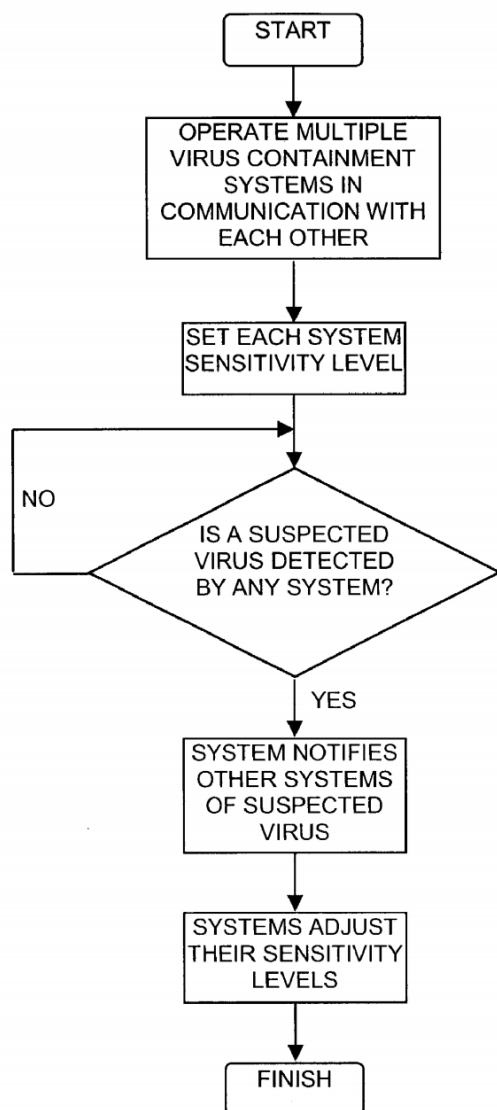


Fig. 7

129. In another example of detecting specific malware, Halperin discloses identifying communication messages that contain copies of the malicious software being transmitted over the network.

... the server may include a buffer or other mechanism whereby messages received from the computer are held, typically for a pre-defined delay period, prior to forwarding the messages to their intended recipients. In this way, should a computer virus send one or more *infected messages* to valid, non-decoy addresses before sending an infected message to a decoy address, *the infected messages to valid, non-decoy addresses that are still*

held at the server may be "quarantined" at the server and thus prevented, together with the infected message to a decoy address, from reaching their intended destinations.
(Ex. 1003 at [0058], emphasis added).

130. A POSITA would have understood “infected” messages to typically include the virus itself in some form or another.

the data including information about the behaviour of the object running on one or more remote computers;

131. The data transmitted to the server described above is explicitly about the behavior of the virus running on the other computers. In the same disclosure, Halperin describes the data received about the virus as “behavior.”

Additionally or alternatively, *server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500* using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, *server 502 may then correlate the presence of target behavior detected at two or more of computers 500* in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers.
(Ex. 1003 at [0047], emphasis added).

132. Thus, I conclude that Halperin discloses limitation 1[b].

3. Limitation 1[c]

133. Limitation 1[c] of the '250 patent reads:

determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;

134. Halperin discloses determining at the centralized server if behavior data received from computers on the network indicates the presence of malicious software.

135. As I explained with reference to limitation 1[b], Halperin uses target behavior to detect computer viruses. Halperin states, “server 502 may then correlate the presence of target behavior detected at two or more of computers 500 in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.***” (Ex. 1003 at [0047], emphasis added). The detecting of malware is necessary step in order to mitigate or stop the malicious software from causing harm. In Halperin, the mitigation is described as virus containment: “Upon detecting a suspicious behavior pattern server 502 may initiate one or more virus containment actions...” (Ex. 1003 at [0057]).

136. Thus, I conclude that Halperin discloses limitation 1[c].

4. Limitation 1[d]

137. Limitation 1[d] of the '250 patent reads:

classifying the computer object as malware when the data indicates that the computer object is malware;

138. Halperin discloses identifying the detected data as malware at the centralized server. I refer to my analysis of limitation 1[b] and 1[c], which I incorporate by reference. In this analysis, Halperin classifies the software as malware as evidenced by the “virus containment” disclosed such as the quarantine process.

139. In more detail, when Halperin detects potential malicious activity, it engages “virus containment” on suspect programs or objects. In doing so, Halperin is identifying the object as malware. For example, Halperin lists the following examples of virus containment in which objects are classified as malware:

- a. Suspending any or all messages sent by computer 100, thereby preventing messages sent by computer 100 from being forwarded to recipients.
 - b. Forwarding messages that are addressed to a decoy address to a third party for analysis, such as a company or other body that produces anti-virus software.
 - c. Notifying a user at computer 100 of the suspicious message activity.
 - d. Notifying a system administrator that a virus may have been detected.
 - e. Stopping all messages from being forwarded by server 108 to their intended destinations. Taking away all privileges that computer 100 has to access network 102 and/or rights to access shared network files or directories.
 - f. Changing the delay period of all messages received by server 108, thus putting the entire network on “virus alert.”;
 - g. Sending a command to network devices connected to network 102, Such as Switches or routers, to block all attempts by computer 100 to access network 102. This may be done, for example, by using SNMP commands.
- (Ex. 1003 at [0024]-[0030]).

140. All of these virus containment elements identify potentially malicious computer objects such as software that sends messages or access the network. A POSITA would have also understood that the messages being analyzed may contain the malware, which is why messages can be analyzed by anti-virus software.

141. Thus I conclude that Halperin discloses limitation 1[d].

5. Limitation 1[e]

142. Limitation 1[e] of the '250 patent reads:

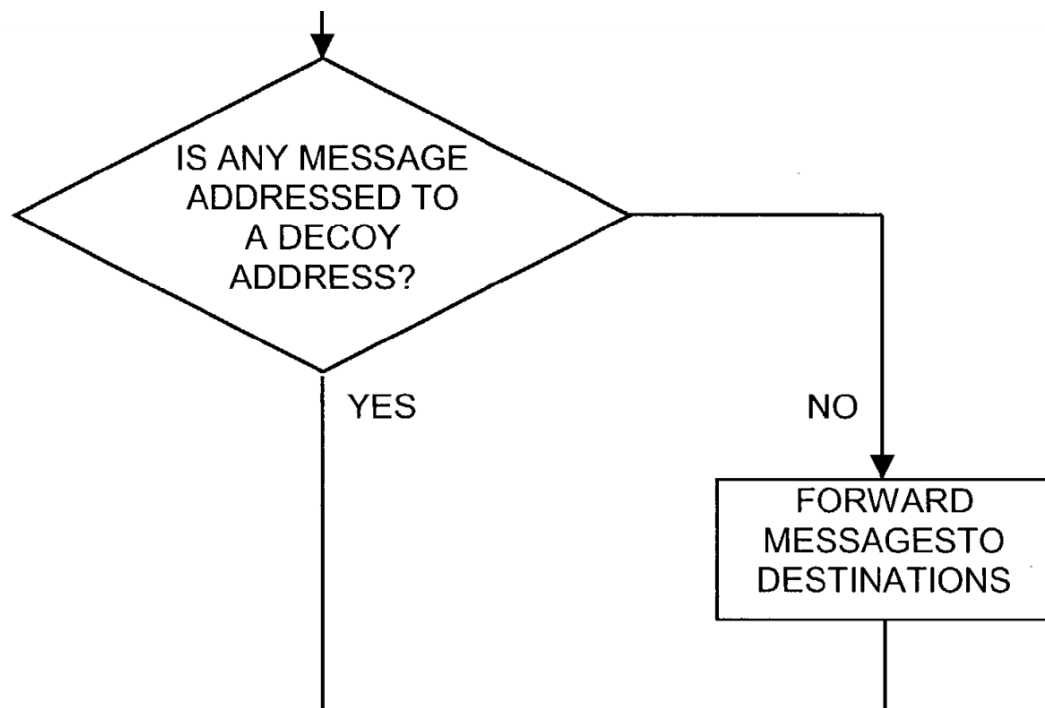
*when the determining does not indicate that the computer object is malware,
initially classifying the computer object as not malware;*

143. Halperin discloses having the centralized server not classifying “suspicious” software as malware in circumstances in which there is not enough evidence to do so. For example, Halperin discusses using a detection phase that analyzes behavior that includes a probability estimate. A customizable threshold determines whether or not to classify the behavior as malicious.

In the detecting phase the system logs every event of a sensor identifying a suspicious behavior on a specific computer into a central database. The logging preferably includes the type of event, the time it occurred and on which computer it occurred (if relevant). Either every predefined period of time or upon each event, the System queries the database for all events that took place during the last time slot. For example, if the time slot is 5 minutes, it gets all the events that took place in the past 5 minutes. These events are grouped by the sensor type and counted for that Sensor type. Thus, for example, it may be found that for the last 5 minute time slot 0 computers sent e-mails to a decoy address, 2 computers accessed a specific key in the Windows Registry and 5 computers accessed a decoy file. ***For each sensor, the probability of getting that result or a higher in a network without a virus is estimated based on the results in the learning database that were analyzed as described above.*** For example, if in the past 5 minutes there were 3 events of accessing the Windows Registry, the system checks what was the measured probability of having 3 or more computers with a Windows Registry event in the learning phase. ***It is then possible to estimate what is the combined probability that the different detectors would have the values that were measured assuming that there is no virus in the system.*** For example, all the results for the detectors for that time slot are converted to probabilities of having that value or higher as described above. The lowest 3 probabilities are taken, and a binomial distribution calculation is done for getting 3 successes out of K attempts when the probability of Success is equal to the highest of the 3 probabilities. ***The result is an estimate of the probability to get the result that was measured assuming that there is no virus in the system. Then the result is compared to a predefined threshold. If the calculated value is lower than the predefined threshold, the system can activate a virus alarm and possibly initiate virus containment actions. The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.*** The system additionally can have an option to have the operator enter information on whether at that time slot there is a centralized operation

that may look like a virus operation, e.g. a mass software installation in the network. In that case the sensors that may be activated by such an operation (e.g. Windows Registry sensors) may be disregarded for the time period defined by the operator for the centralized operation.
(Ex. 1003 at [0090], emphasis added)

144. An another example, Halperin has decision trees at the centralized server that categorize objects as either malicious or not malicious. For example, messages forwarded to a “decoy” address can be categorized as malicious and otherwise are categorized as not malicious. This process is visualized in Figure 2, reproduced hereinbelow.



145. A POSITA would have understood that messages forwarded to destinations are **not** categorized as malware.

146. Thus, I conclude that Halperin discloses limitation 1[e] .

6. Limitation 1[f]

147. Limitation 1[f] of the '250 patent reads:

automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data;

148. Halperin discloses having the centralized server identifying acceptable behavior for software running on the monitored computers. Mankins discloses automatically creating a bloom filter that identifies normal operations for the software object based on training information. The bloom filter acts as the “mask” required by this limitation, as explained in more detail below.

149. Halperin discloses automatically monitoring system events and using “retrospective” information about known virus outbreaks to events that could be related to the virus and events that are not. A POSITA would have understood Halperin’s disclosure to refer to potentially unacceptable behavior (related to the virus) and acceptable behavior (when no viruses are present). Halperin explains:

In the learning phase ***all events are logged*** in a learning database similar to the database described above. This is done for a substantial period of time, for example, one week. In addition, the system gets retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system...

The learning database is sorted according to the time of each event logged. The events are grouped into non-overlapping time slots of a predefined length (e.g. 5 minutes). ***The learning database is then filtered*** according to the data logged in the virus outbreak database. ***Any time slot that is logged in the virus outbreak database as having an active virus in the system is taken out of the learning database.***

(Ex. 1003 at [0087]-[0088], emphasis added)

150. Mankins discloses a similar learning process for a specific piece of software. The software’s “normal” operations (e.g., system calls to the OS) are automatically learned during a training phase. During the training session, the calls to the operating system are monitored and recorded. The monitored data is encoded into a bloom filter that is associated with the software

object (e.g., appended to it). When the program is running, its behavior is compared to the bloom filter. If the program attempts to deviate from the “trained” behavior, the system raises an alert.

To define normal behavior, a particular computer program 130 may be invoked and operated ***under training conditions*** in order to determine a set of operations initiated by the computer program 160.
(Ex. 1004 at 6:63-66, emphasis added).

The build subsystem 160 is configured to ***store*** the generated profiles of normal program behavior (e.g., sets of unique system-call sequences) compactly by using ***[a] Bloom filter*** 175.
(Ex. 1004 at 7:22-25, emphasis added).

The build subsystem 160 may be configured to associate a generated Bloom filter data object 180 to its corresponding computer program 130. In particular, the build subsystem 160 may cause the Bloom filter data object 180 to be ***appended to the executable image*** of the computer program 130 in a manner that will cause the Bloom filter data object 180 to be automatically embedded in the OS kernel 120 upon invocation of the computer program 130.
(Ex. 1004 at 7:59-66, emphasis added).

The intrusion detection system 100 may be configured to ***automatically*** generate Bloom filter data objects 180 for computer programs 130 and then ***automatically*** use the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time.
(Ex. 1004 at 8:8-13, emphasis added).

151. Thus, Mankins’ bloom filter storing normal (non-malicious) operations maps precisely with the concept of the mask as explained in the ’250 patent. As stated in the specification:

If an object is allowed to run on a remote computer 2... then operation of that object may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on. Thus, the mask can give an overall picture of the expected “normal behaviour” of that object.

(Ex. 1001 at 14:58-15:4).

152. Mankins teaches that “a profile of normal program behavior” thus generated is contained in a “data object”:

As known to those skilled in the art, Bloom filters 175 include data structures that are well-suited for compactly storing data for use in membership tests. In particular, the build subsystem 160 may use the Bloom filter 175 to generate a particular Bloom filter data object 180 from a profile of normal program behavior. Accordingly, ***the generated Bloom filter data object 180 will contain data representative of the normal behavior expected of a particular computer program 130.***

(Ex. 1004 at 7:25-33, emphasis added).

153. Mankins further explains, “‘Normal’ behavior may be defined differently but generally refers to observable and ***acceptable behavior characteristics*** expected of networks when not under attack.” (Ex. 1004 at 1:41-43, emphasis added). It would have been obvious to a POSITA to automatically generate Mankins’ bloom filter of “normal program behavior” (e.g., “acceptable behavior characteristics”) on the central system after receiving information about the object from the remote computers disclosed by Halperin. Halperin relies on data from multiple systems that is transmitted to and received by the central computer in order for the central computer to determine if the software is good or bad. Moreover, because the software has appeared on multiple computers, it is most efficient to generate the bloom filter once on the central computer and then transmit the protected software (e.g., with the bloom filter appended to the file) out to all the computers that need to run the program.

154. A POSITA would also have noted that Mankins explicitly promotes a “distributed” environment as an option: “...the intrusion detection system 100 may be implemented in a distributed fashion over a network of computing devices.” (Ex. 1004 at 4:29-31). A POSITA would have known that a “distributed” system uses data spread out over many components; however, the actual generation of the bloom filter would generally have to be centralized to one

component. Thus, a POSITA would have known that training data can be extracted from anywhere within the distributed system (e.g., running on multiple computers) but that the bloom filter would most likely be produced within a single (centralized) system. At the very least, it would have been obvious to a POSITA to combine Mankins with Halperin this this fashion.

155. A POSITA would have been motivated to combine Halperin and Mankins. As explained in Section IX.A., a POSITA would be motivated to combine these two references in general. But specifically, it would have been obvious use Mankins as described in this section to build a bloom filter of normal behavior with Halperin in order to extend Halperin's dynamic system. That is, Halperin discloses a system that learns as it goes, collecting data about "normal" and "abnormal" operations. Halperin, for example, discloses using increasing suspicious activity to eventually activate containment protocols.

In an alternative implementation of the buffer described above that is designed to reduce false alarms, should the server receive an invalid decoy message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified. During the increased delay period, ***should additional suspicious messages be received, or should other suspicious behavior be detected, if the user and/or system administrator who is authorized to do so has not indicated that the activity is not virus related, only then does the server perform one or more virus containment actions.*** If, however, during the increased delay period no other suspicious activity is detected, or if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.
(Ex. 1003 at [0059], emphasis added.)

156. Nevertheless, Halperin does identify that one of the problems of this approach is that there must be a trade off between false alarms (detecting a "good" program as a bad one) and not detecting malware at all. As Halperin explains, "... the [probability] result is compared to a predefined threshold. If the calculated value is lower than the predefined threshold, the system can

activate a virus alarm and possibly initiate virus containment actions. ***The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.***” (Ex. 1003 at [0090], emphasis added).

157. Mankins offers another alternative to a simple false-alarm/less-detection trade-off. A program believed to be benign can have the bloom filter of trained data attached to it limiting the software to its “good” behavior. That way, if the software “gets worse,” it is stopped dynamically. As Mankins explains, “Preferably, high-quality hash functions are used in the Bloom filter 175 to help guarantee that output is equally distributed over all possible values in the bit vector, which helps to minimize the occurrence of false positives.” (Ex. 1004 at 8:38-41).

158. A POSITA would have been motivated to add these protections to Halperin, preventing damage when malware is not detected and improving the false positive rates.

159. Thus, I conclude that Halperin in view of Mankins discloses limitation 1[f].

7. Limitation 1[g]

160. Limitation 1[g] of the ’250 patent reads:

running said object on at least one of the remote computers;

161. Halperin discloses allowing software to run on remote computers until certain triggers are reached (e.g., a probability threshold exceeded). I discussed this disclosure in context of limitation 1[f] and I incorporate my analysis of this disclosure into this section by reference.

162. Mankins discloses a bloom filter that allows software to run within a monitoring environment that enforces acceptable behavior based on training data. I discussed this disclosure in context of limitation 1[f] and I incorporate my analysis of this disclosure into this section by reference.

163. A POSITA would have been motivated to use Mankins' bloom filter of "normal" behavior to protect systems from software running on the remote computers that, although not yet determined to be viruses, had suspicious triggers. As I explained in reference to limitation 1[f], Halperin allows such software to run until it has a high enough confidence to decide to block it. The confidence required is configurable and the trade-off is false alarms for lack of detection. Using Mankins, it is possible to allow the software to run with a profile of acceptable behavior. If any activities stray from this profile, the execution is stopped preventing damage to the remote systems. As Mankins explains, "Preferably, high-quality hash functions are used in the Bloom filter 175 to help guarantee that output is equally distributed over all possible values in the bit vector, which helps to minimize the occurrence of false positives." (Ex. 1004 at 8:38-41).

164. Thus, I conclude that Halperin in view of Mankins limitation 1[g].

8. Limitation 1[h]

165. Limitation 1[h] of the '250 patent reads:

automatically monitoring operation of the object on the at least one of the remote computers;

166. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

167. Mankins discloses automatically monitoring software as it runs to detect if its behavior strays outside of an authorized profile of approved operations. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by

reference. I also explained why a POSITA would have been motivated to use Mankins' disclosures in combination with Halperin's teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

168. Thus, I conclude that Halperin in view of Mankins discloses limitation 1[h].

9. Limitation 1[i]

169. Limitation 1[i] of the '250 patent reads:

allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;

170. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

171. Mankins discloses automatically monitoring software as it runs to detect if its behavior strays outside of an authorized profile of approved operations. Mankins discloses a bloom filter of normal operations as a means to enable a monitoring subsystem to detect anomalous operations. If such operations are detected, it sends an alert. "At step 340, an alert is issued in response to a determination that the Bloom filter data object 180 does not contain a match for the monitor behavior. As mentioned above, the alert may be issued to the operating system or user of the computer 110." (Ex. 1004 at 10:7-10). Mankins also explains that an alert can trigger any predefined operation. "Any appropriate predefined action may be performed in response to the alert." (Ex. 1004 at 10:13-14). A POSITA would have known that shutting down a suspicious program executing outside of its training operations is one such "appropriate predefined action."

Obviously, when the behavior is within the scope of the bloom filter, there is no alert and the operations of the program continue normally.

172. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference. I also explained why a POSITA would have been motivated to use Mankins' disclosures in combination with Halperin's teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

173. Thus, I conclude that Halperin in view of Mankins discloses limitation 1[i].

10. Limitation 1[j]

174. Limitation 1[j] of the '250 patent reads:

disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and

175. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

176. Mankins discloses automatically monitoring software as it runs to detect if its behavior strays outside of an authorized profile of approved operations. Mankins discloses a bloom filter of operations detected during training as a means to enable this monitoring. If anomalous operations are detected, the monitoring system sends an alert. "At step 340, an alert is issued in response to a determination that the Bloom filter data object 180 does not contain a match for the monitor behavior. As mentioned above, the alert may be issued to the operating system or user of the computer 110." (Ex. 1004 at 10:7-10). Mankins also explains that an alert can trigger

any predefined operation. “Any appropriate predefined action may be performed in response to the alert.” (Ex. 1004 at 10:13-14). A POSITA would have known that shutting down a suspicious program executing outside of its training operations is one such “appropriate predefined action.”

177. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference. I also explained why a POSITA would have been motivated to use Mankins disclosures in combination with Halperin’s teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

178. Thus, I conclude that Halperin in view of Mankins discloses limitation 1[j].

11. Limitation 1[k]

179. Limitation 1[k] of the ’250 patent reads:

reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

180. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. In other words, Halperin initially classifies the software as **not malicious** but as more information is received (especially from multiple computers) it changes its classification to “virus.” I incorporate my analysis of this disclosure into this section by reference.

181. Mankins discloses automatically monitoring software as it runs to ensure that its behavior does not stray outside of an authorized profile of approved operations. Mankins discloses a bloom filter of acceptable operations learned during training as a means to enable this monitoring

and to shut down the software if it misbehaves. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference.

182. A POSITA would have been motivated to combine Mankins' bloom filter of normal behavior with Halperin such that if the monitoring system detected misbehavior it would alert Halperin's centralized server. As I have already explained, Halperin's central server collects data about behavior that is deemed suspicious. Halperin explains that it uses many "sensors" (each sensor typically running on one of the computers in the network) to collect the data: "In the embodiments described hereinabove for detecting suspicious activities in a computer network different detectors (e.g. file deletion and Windows Registry access) and decoys (e.g. e-mail decoy) were defined. These will all be referred to from this point onward as sensors." (Ex. 1003 at [0086].)

183. It would have been obvious to a POSITA to use Mankins monitoring of the bloom filter as a "detector" or "sensor." A POSITA would have recognized that behavior that goes outside of the bloom filter's approved profile is suspicious and Halperin's central server would be interested in the information. As Halperin explains, for "...the [malware] detecting phase the system logs **every event** of a sensor identifying a suspicious behavior on a specific computer into a central database." (Ex. 1003 at [0090], emphasis added.) As explained with reference to limitation 1[f], Mankins explicitly sends alerts when behavior goes outside of the bloom filter.

184. As also explained previously, once Halperin has received additional suspicious information about software it will recategorize it as a virus and execute containment protocols.

185. Thus, I conclude that Halperin in view of Mankins discloses limitation 1[k].

C. Claim 2 is rendered obvious by Halperin in view of Mankins

186. Claim 2 of the '250 patent, which depends from Claim 1, reads:

A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of: executable

instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

187. I have already explained that Halperin in view of Mankins render obvious Claim 1. Halperin discloses the additional requirements of Claim 2.

188. Halperin discloses that the target behavior, which is data about the potential virus sent from multiple computers to a central server, includes:

- a. Sending messages to more than a predefined number of users during a predefined time.
- b. Sending messages not as a result of a direct user interaction with the Graphic User Interface (GUI) of the message software, but rather as the result of a directive from a software application.
- c. Modifying operating system files such as the Microsoft Windows registry.
- d. Deleting more than a predefined number of files on the computer's hard disk during a predefined period of time.
- e. Loading a new software application into the computer's RAM.
- f. Sending a file attached to a message several times from the same user.
- g. Sending a file attachment of a specific type (e.g., .exe, .doc, zip).
- h. Attempting to contact previously unused or unknown IP addresses or IP sockets.

(Ex. 1003 at [0038]-[0046]).

189. All of these behaviors are examples of events initiated by the object when the object is running on the respective remote computers.

190. Thus, I conclude that Halperin discloses the additional requirements of Claim 2.

D. Claim 3 is rendered obvious by Halperin in view of Mankins

191. Claim 3 of the '250 patent, which depends from Claim 1, reads:

A method according to claim 1, wherein determining identifies relationships between the object and other objects.

192. I have already explained that Halperin in view of Mankins render obvious Claim 1. Halperin discloses the additional requirements of Claim 3.

193. Halperin discloses having the centralized server determine if software is potentially a virus by correlating together data about the behavior of software across multiple machines and its interaction with other software and systems. Halperin states:

After collecting information regarding target behavior detected at two or more of computers 500, server 502 may then correlate the presence of target behavior detected at two or more of computers 500 in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers. Any known behavior correlation techniques may be used, such as identifying the same activity in different computers at about the same time, or by identifying repeating patterns of data within the memories of two or more computers.
(Ex. 1003 at [0047]).

194. Halperin provides multiple examples of identifying “suspicious behavior patterns” across multiple machines (*Id.*):

- a. A certain percentage of the computers in the network sending more than 10 messages per minute in the last 5 minutes,

- b. A certain percentage of the computers in the network sending messages not initiated via the message GUI in the last 1 minute;
- c. A certain percentage of the computers in the network deleting more than 10 files in the last 1 minute;
- d. A certain percentage of computers in the network deleting a file by the same name within the last 1 hour.
- e. A certain percentage of the computers in the network deleting a file with the same name in the last 1 minute;
- f. A certain percentage of the computers in the network to which changes to the Microsoft Windows Registry occurred in the last 1 minute;
- g. A certain percentage of the computers in the network sending the same file attachment via a message in the last 15 minutes,
- h. A certain percentage of the computers in the network sending file attachments via one or more messages in the last hour where each of the files includes the same string of bits,
- i. A certain percentage of the computers in the network having an unusual level of correlation of data between files sent as attachments. For example, since viruses known as “polymorphic viruses” may change their name as they move from one computer to another, one way to identify such viruses is to identify attachments that have the same or similar data, whether or not they have the same name.

(Ex. 1003 at [0048]-[0056]).

195. A POSITA would have understood that the suspicious behavior patterns, such as the explicit examples listed above, identify relationships between the potential malware, which is “the” computer object as discussed in limitation 1[b], and other objects such as messages, files, attachments, registry keys, other programs, and other computers. I note that, although the example patterns above do not explicitly name the potential malware, Halperin is explicit that the potential malware must be identified (at the server) as part of the pattern process. As I explained with respect to limitation 1[b], Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.*** (Ex. 1003 at [0047], emphasis added).

196. I provided additional analysis with respect to limitation 1[b] and that Halperin identifies the malware that causes the suspicious behaviors. I incorporate this analysis into this section by reference.

197. Thus, for each of the patterns described in Halperin, a POSITA would have recognized that the potential malware, which is “the” computer object, is related to the other computer objects by the nature of the pattern.

198. For example, a POSITA would have recognized that a pattern of “[a] certain percentage of the computers in the network sending more than 10 messages per minute in the last 5 minutes” identifies the relationship between the potential malware sending the messages on the computers and the computers receiving the messages, which are “other objects.” In particular, a

POSITA would classify the relationship as “repeatedly-transmits-to.” That is, the potential malware’s relationship to the other computers is one of a “repeated-transmitter.”

199. Similarly, a POSITA would have recognized that a pattern of “[a] certain percentage of the computers in the network sending messages not initiated via the message GUI in the last 1 minute” identifies the relationship between the potential malware and other computers on the network as “automatically-and-repeatedly-transmits-to.” That is, the potential malware’s relationship to the other computers is one of an “automated-and-repeated-transmitter.”

200. Additionally, a POSITA would have recognized that a pattern of “[a] certain percentage of the computers in the network deleting more than 10 files in the last 1 minute” identifies the relationship between the potential malware and files on computers on the network as “rapidly-deletes.” That is, the potential malware’s relationship to the other files (which are “other objects”) is one of “rapid-deleter-of.”

201. As with these three illustrations, each of the other suspicious patterns identifies a relationship between the potential malware and other computer objects.

202. The disclosures in Halperin mirror the examples provided in the specification of the ’250 patent. The specification explains that events “define the actions or behaviours of an object acting upon another object or some other entity.” (Ex. 1001 at 12:64-13:5). The events identify the acting object (e.g., a program), the action itself, and the target or “victim” of the action. (*Id.*). The specification provides the following table as an example of these kinds of events.

Actor	Event Type	Victim
Object 1	Creates Program	Object 2
Object 1	Sends data	IP Address 3
Object 1	Deletes Program	Object 4
Object 1	Executes	Object 2
Object 2	Creates registry key	Object 4

(*Id.* at 13:8-15).

203. The specification goes on to explain that **from** these events, “this structure allows a limitless series of behaviours and relationships to be defined.” (*Id.* at 12:64-13:5). Similarly, a POSITA would have recognized that Halperin, using a nearly identical event structure, defines a series of pattern-based relationships.

204. Moreover, Halperin identifies related objects across computers based on correlated suspicious activity. “Peer-to-peer communication techniques may be used to transfer information within the group, and the correlation calculation may be performed by any of the computing device peers. A similar process may be implemented within neighboring groups to allow correlation of suspicious activities between groups.” (Ex. 1003 at [0078]).

205. Thus, I conclude that Halperin discloses the additional requirements of Claim 3.

E. Claim 4 is rendered obvious by Halperin in view of Mankins

206. Claim 4 of the '250 patent, which depends from Claim 3, reads:

A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

207. I have already explained that Halperin in view of Mankins render obvious Claim 3. Halperin discloses the additional requirements of Claim 4.

208. Halperin discloses sharing information between computers, and aggregated at the central server, to identify malware spread from one computer to another. For example, Halperin discloses using attachments that are identified as viruses to identify other attachments as viruses. “For example, since viruses known as ‘polymorphic viruses’ may change their name as they move from one computer to another, one way to identify such viruses is to identify attachments that have the same or similar data, whether or not they have the same name.” (Ex. 1003 at [0056]). These

attachment objects are related to each other by having the same or similar data. Identifying one as malware is used to identify the others as malware.

209. In another example, Halperin discloses that identifying a characteristic of malware at the centralized server in one set of computers can be used to identify that malware in another group of computers. According to Halperin, “if a change to a certain software variable is used to identify the malicious software in one group, the same change may be monitored for in neighboring groups.” (Ex. 1003 at [0069]).

210. In a third example, Halperin also explains that a “checksum” (which is a code that identifies data) is generated for malware on one machine, that same code can be forwarded to other machines for use in identifying the same malware. “A characteristic code may be generated for the attachment that distinguishes it from other such attachments. This may be done using well known ‘checksum’ algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the checksum to identify suspicious malicious software upon arrival at these computers.” (Ex. 1003 at [0073]).

211. A POSITA would recognize that Halperin’s teachings with respect to “checksums” identifying suspicious malicious software are similar to the teachings of the ’250 patent specification on that point:

Referring now to FIG. 2, at the start point 21, a computer object 4 such as a process is run at a remote computer 2. At step 22, by operation of local “agent” software running on the remote computer 2, the operation of the process is hooked so that the agent software can search a local database stored at the remote computer 2 ***to search for a signature or key representing that particular process, its related objects and/or the event.*** If the local signature is present, it will indicate either that the process is considered to be safe or will indicate that that process is considered unsafe. An unsafe process might be one that has been found to be malware or to have unforeseen or known unsafe or malevolent results arising from its running. If the signature indicates that the process is safe, then that process or event is allowed by the local agent software on the remote computer 2 to

run at step 23. If the signature indicates that the process is not safe, then the process or event is stopped at step 24.
(Ex. 1001 at 8:17-34, emphasis added).

212. In all of these examples, Halperin discloses the use of one identified virus/malware to identify the same or similar viruses on other machines.

213. Thus, I conclude that Halperin discloses the additional requirements of Claim 4.

F. Claim 5 is rendered obvious by Halperin in view of Mankins

214. Claim 5 of the '250 patent, which depends from Claim 3, reads:

A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

215. I have already explained that Halperin in view of Mankins render obvious Claim 3. Halperin discloses the additional requirements of Claim 5.

216. As I explained with reference to Claim 3, Halperin discloses identifying at the centralized server the malware/virus objects on the remote computers. A POSITA would have understood that this means “stored” on the remote computers. I incorporate my analysis of Claim 3 into this section by reference.

217. Moreover, Halperin identifies the same or similar objects stored across the remote computers based on correlated suspicious activity. “Peer-to-peer communication techniques may be used to transfer information within the group, and the correlation calculation may be performed by any of the computing device peers. A similar process may be implemented within neighboring groups to allow correlation of suspicious activities between groups.” (Ex. 1003 at [0078]).

218. Thus, I conclude that Halperin discloses the additional requirements of Claim 5.

G. Claim 12 is rendered obvious by Halperin in view of Mankins

219. Claim 12 of the '250 patent, which depends from Claim 1, reads:

A method according to claim 1, comprising if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future.

220. I have already explained that Halperin in view of Mankins render obvious Claim 1. Halperin in view of Mankins discloses the additional requirements of Claim 12.

221. Halperin discloses that a fundamental part of the system is to monitor **at the centralized server** suspicious behavior of remote computers and use detected suspicious behavior to identify viruses. Halperin also recognizes that, in order to reduce false alarms, there must be a means to “clear” the behavior of being suspicious, which is also performed at the centralized server. One means of doing this is simply to have an administrator of the central server indicate that the activity is safe. Halperin explains that “...to reduce false alarms, should **the server** receive an invalid decoy message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified... if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.” (Ex. 1003 at [0059], emphasis added).

222. Halperin also teaches that quarantines provide an opportunity for a “suspicious” object to be cleared, which also takes place at the central server. “**The server** may also notify a system administrator of the quarantined messages who may then check the quarantined to

determine whether or not the messages were indeed sent by a computer virus and either allow them to be forwarded to their intended recipients as is..." (Ex. 1003 at [0058], emphasis added).

223. I note that these operations in Halperin are driven by decisions made by a human user. While certainly partially automated, an administrator must decide whether to clear an "alarm" level or to release something from quarantine. Similarly, a POSITA would have known that the requirements of this Claim could involve at least some human intervention. For example, a POSITA would have known that decisions made with respect to allowing a "misbehaving" file to be re-evaluated could take advantage of human interactions. That is, if an automated system is going to be overridden, it can be done by a human. Moreover, selecting the objects to compare to could involve some kind of human intervention. Thus a POSITA would not read this claim in such a way as to permit, full automation, even if a fully automated solution is one possibility.

224. As I noted above, in addition to identifying suspicious behavior, Halperin also discloses identifying normal behavior at the central server. Learning "normal" behavior is necessary for identifying "abnormal" behavior. Halperin teaches that "[o]nce the groups are defined, it is possible to define and measure different parameters that are indicative of the methods of operation within and between the groups. Over time the characteristic values of these parameters during normal operation may be learned." (Ex. 1003 at [0071]). A POSITA would have understood that this process can permit a human to be involved to assist the system to continue to learn correctly.

225. It would have been obvious to a POSITA to also use the behavior of objects to ***continuously*** identify normal patterns of behavior and not just once. In other words, a POSITA would have found it obvious to use on-going data received at the central server to retrain the system, as directed by the administrator, as new information becomes available. As noted above,

this is already part of Halperin's design and it would have been obvious to incorporate Mankins in the same way.

226. As noted previously, Mankins discloses a learning a software's "normal" operations (e.g., system calls to the OS) during a training phase. During this phase, the calls to the operating system are monitored and recorded. The monitored data is encoded into a bloom filter that is associated with the software object (e.g., appended to it). When the program is running, its behavior is compared to the bloom filter. If the program attempts to deviate from the "trained" behavior, the system raises an alert.

To define normal behavior, a particular computer program 130 may be invoked and operated ***under training conditions*** in order to determine a set of operations initiated by the computer program 160.
(Ex. 1004 at 6:63-66, emphasis added).

The build subsystem 160 is configured to ***store*** the generated profiles of normal program behavior (e.g., sets of unique system-call sequences) compactly by using ***[a] Bloom filter*** 175.
(Ex. 1004 at 7:22-25, emphasis added).

The build subsystem 160 may be configured to associate a generated Bloom filter data object 180 to its corresponding computer program 130. In particular, the build subsystem 160 may cause the Bloom filter data object 180 to be ***appended to the executable image*** of the computer program 130 in a manner that will cause the Bloom filter data object 180 to be automatically embedded in the OS kernel 120 upon invocation of the computer program 130.
(Ex. 1004 at 7:59-66, emphasis added).

The Intrusion detection system 100 may be configured to ***automatically*** generate Bloom filter data objects 180 for computer programs 130 and then ***automatically*** use the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time.
(Ex. 1004 at 8:8-13, emphasis added).

227. It would have been obvious to a POSITA to update the bloom filter of normal operations at the central server as part of the continuous retraining discussed above. A POSITA

would have found it obvious to “quarantine”, at the central server, programs that triggered alerts from Mankins monitoring. And, in fact, Mankins explicitly suggests that the alert may be sent to a user (e.g., administrator) of the violation, It would have been equally obvious that if the administrator would use continuous training to update the bloom filter if the software was not, in fact, malicious. As discussed above, the administrator disclosed by Halperin controls and monitors the network from a central server.

228. Halperin also teaches that a network with detection of suspicious behavior may be divided into a “live set” and a “test set.” Bad behavior detected at the “live set” will result in the virus containment procedures I have discussed in previous limitations. The “test set,” on the other hand, is used for monitoring and learning about the malware so that true viruses or false positives can more thoroughly be detected. As Halperin explains:

Any of groups 800 may have different sets of sensors, ***such as one live set and one test set.*** “Different set of sensors” may actually be different types of sensors, different thresholds for similar sensors, or different algorithms to identify Suspicious activity based on the gathered data. ***The live set is used for implementation of virus containment protocols*** as described hereinabove, ***while the test set monitors for malicious software and logs the results in order to test new sensor and correlation algorithms.*** Live and test set responses to system events, such as actual virus detections and false alarms, ***may be compared to identify algorithm effectiveness.*** This may be performed retrospectively ***once a series of system alerts have been identified as either real virus alerts or false alarms.***
(Ex. 1003 at [0075], emphasis added).

229. As Halperin explains, the purpose of the test set is to test sensor and correlation algorithms. The use of both a live set and a test set is used to identify algorithm effectiveness. (*Id.*). In other words, the suspicious behaviors and the responses thereto can be compared between different groups of computers in order to determine correct responses including true virus identification rather than false alarms.

230. A POSITA would have found it obvious based on Halperin's teachings in view of Mankins' teachings that the masked processes discussed above be compared against equivalent processes with different masks in different sets of groups (e.g., "live set" vs "test set" groups). A POSITA would have been motivated to use Halperin's teachings that the different groups of computers may have "different thresholds" to Mankins' bloom filters, which are the claimed "masks", such that the same software operates with different bloom filters ("masks") with different limitations on a test set vs a live set of computers. As Halperin explains, this permits the administrator to determine algorithm/threshold effectiveness. This process may be retrospective, as taught by Halperin, based on new information about true viruses and false positives.

231. A POSITA would have understood this process to involve comparing the behavior of software that triggers alerts based on their bloom filters, which is "monitored behaviour of the object running on the remote computer" with the same behavior of the same software with different bloom filters that did not report alerts, which is "comparing at the base computer the non-permitted behaviour with the behaviour of other objects". If the behavior of the non-alerting objects is determined to be safe the behavior of the alerting objects will also be permitted. And, as described above, the mask will be retrained to permit the behavior going forward. In other words, "if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future," as required by the claim.

232. It would have been obvious to combine Halperin and Mankins as discussed in Section IX.A. A POSITA would have recognized that the use continuous training of Mankins in combination with Halperin's system would further reduce false positives and further enable the administrator to respond to identified false positives.

H. Claim 13 is rendered obvious by Halperin in view of Mankins

233. Halperin in view of Mankins discloses all of the limitations of Claim 13.

1. Limitation 13[a]

234. Limitation 13[a] of the '250 patent reads:

Apparatus for classifying a computer object as malware, the apparatus comprising

235. See analysis of limitation 1[a].

236. Halperin's discloses an apparatus. Specifically, Halperin's "computer virus detection system" is an apparatus as shown in the Figure 1 and Figure 5 reproductions below (see also Ex. 1003 at [0036]):

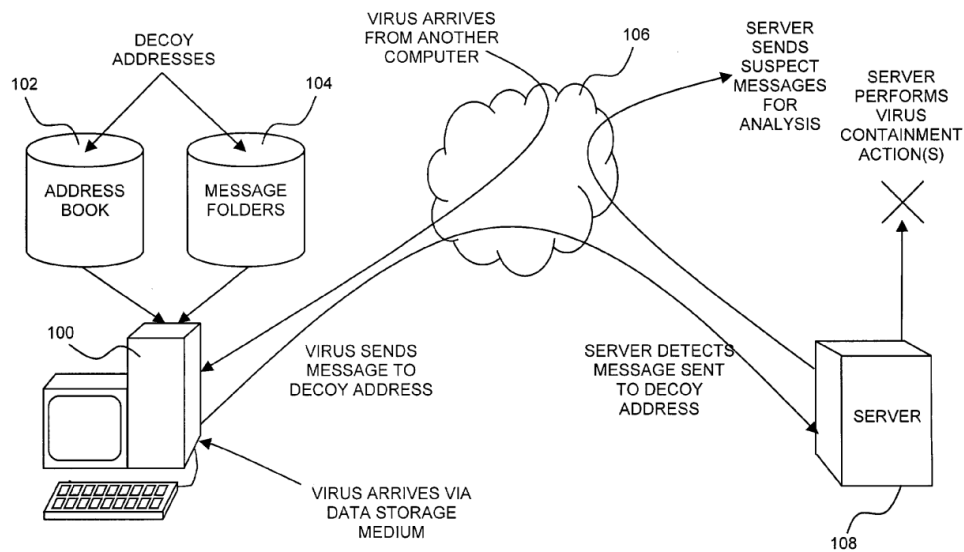


Fig. 1

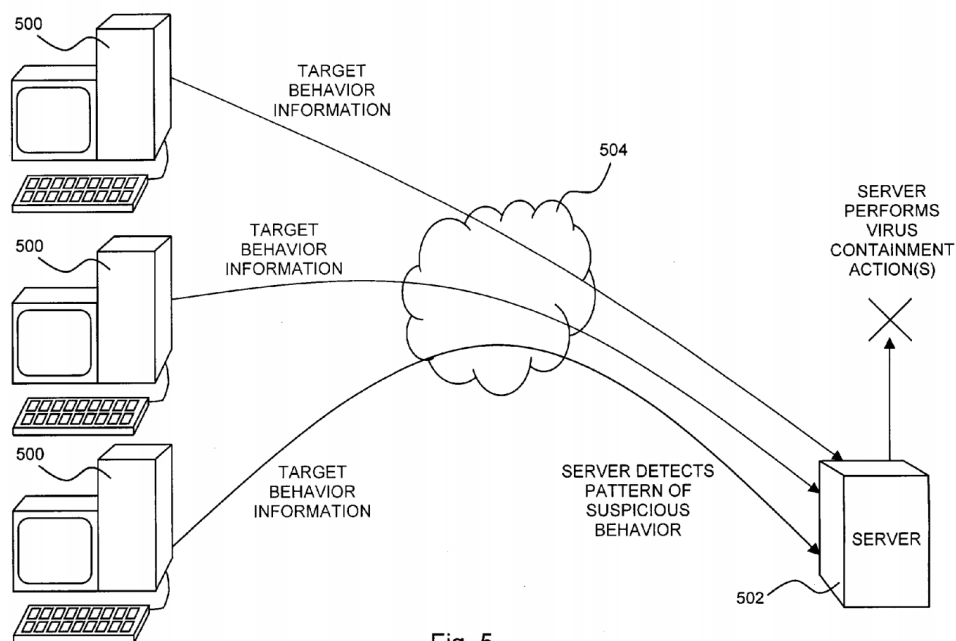


Fig. 5

237. Moreover, Halperin discloses, “While the methods and apparatus disclosed herein may or may not have been described with reference to specific hardware or software, it is appreciated that the methods and apparatus described herein may be readily implemented in hardware or software using conventional techniques.” (Ex. 1003 at [0092]).

238. Thus, Halperin discloses limitation 13[b].

2. Limitation 13[b]

239. Limitation 13[b] of the '250 patent reads:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

240. See analysis of limitation 1[b]. As I explained with reference to Claim 1, Halperin discloses a centralized server that functions as a “base computer” performing the operations recited

in Claim 13 and its dependent claims. The centralized server is thus “constructed and configured” to perform the operations of these claims.

241. Thus, I conclude that Halperin discloses limitation 13[b].

3. Limitation 13[c]

242. Limitation 13[c] of the '250 patent reads:

the base computer being constructed and arranged to determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,

243. See analysis of limitation 1[c].

244. Thus, I conclude the Halperin discloses limitation 13[c].

4. Limitation 13[d]

245. Limitation 13[d] of the '250 patent reads:

the base computer being constructed and arranged to classify the computer object as malware when the base computer determines that the data indicates that the computer object is malware,

246. See analysis of limitation 1[d].

247. Thus, I conclude that Halperin discloses limitation 13[d].

5. Limitation 13[e]

248. Limitation 13[e] of the '250 patent reads:

wherein the base computer is constructed and arranged to automatically generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data,

249. See analysis of limitations 1[e]-1[f].
250. Thus, I conclude that Halperin in view of Mankins discloses limitation 13[e].

6. Limitation 13[f]

251. Limitation 13[f] of the '250 patent reads:

wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers, the object being allowed to continue to run when behaviour of the object is permitted by the mask, the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.

252. See analysis of limitations 1[g]-1[k].
253. Thus, I conclude that Halperin in view of Mankins discloses limitation 13[f]

I. Claim 14 is rendered obvious by Halperin in view of Mankins

254. Claim 14 of the '250 patent, which depends from Claim 13, reads:

Apparatus according to claim 13, the data includes one or more of executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

255. See analysis of Claim 2.

256. Thus, I conclude that Halperin in view of Mankins discloses Claim 2.

J. Claim 15 is rendered obvious by Halperin in view of Mankins

257. Claim 15 of the '250 patent, which depends from Claim 13 reads:

Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the determining identifies relationships between the object and other objects.

258. See analysis of Claim 3.

259. Thus, I conclude that Halperin in view of Mankins discloses Claim 15.

K. Claim 16 is rendered obvious by Halperin in view of Mankins

260. Claim 16 of the '250 patent, which depends from Claim 15 reads:

Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

261. See analysis of Claim 4.

262. Thus, I conclude that Halperin in view of Mankins discloses Claim 16.

L. Claim 17 is rendered obvious by Halperin in view of Mankins

263. Claim 17 of the '250 patent, which depends from Claim 15, reads:

Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least Sonic of said remote computers.

264. See analysis of Claim 5.

265. Thus, I conclude that Halperin in view of Mankins discloses Claim 17.

M. Claim 24 is rendered obvious by Halperin in view of Mankins

266. Claim 24 of the '250 patent, which depends from Claim 13, reads:

A method according to claim 13, wherein, if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, the base computer is arranged to compare the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects the base computer is arranged to allow said behaviour for the object and to modify the mask to allow that behaviour for that object in future

267. See analysis of Claim 12.

268. Thus, I conclude that Halperin in view of Mankins discloses Claim 24.

X. GROUND 2: CLAIMS 6 AND 18 ARE RENDERED OBVIOUS BY HALPERIN IN VIEW OF MANKINS IN FURTHER VIEW OF ARNOLD

269. As explained in the sections and paragraphs below, Halperin in view of Mankins in further view of Arnold renders obvious the identified claims.

A. Motivation to Combine Halperin and Mankins and Arnold

270. A POSITA would have been motivated to combine Halperin, Mankins, and Arnold. I have already explained in Section IX.A. that it would have been obvious to combine Halperin and Mankins. I incorporate this analysis by reference.

271. A POSITA would have been further motivated to combine Halperin and Mankins with Arnold. Both Halperin and Arnold are also directed to the same problem. That is both Halperin and Arnold are directed to protecting a network from intrusion and especially from malware (such as viruses). Both Halperin and Arnold are concerned with dealing with previously unknown viruses. Mankins is also directed toward computer security. As I explained previously, Halperin provides reactive security while Mankins provides proactive security. Arnold, as I will explain below, combines a bit of both approaches and enhances some of the security goals of both.

272. With respect to Halperin and Arnold, both have a reactive component that examines the system for unknown viruses. Halperin for example, states, “most virus scanners operate by searching a computer for tell-tale byte sequences known as ‘signatures’ that exist in known viruses. Thus, by definition, new viruses whose byte sequences are not yet known to virus scanners cannot be detected in this manner... The present invention seeks to provide for the detection and containment of malicious computer programs that overcomes [this].” (Ex. 1003 at [0005]-[0007]).

273. Arnold also is directed toward reactively identifying previously unknown viruses. “It is further object of this invention to provide methods and apparatus for immunizing a computer system, and also a network of computer systems, against a subsequent infection by a previously unknown and undesirable software entity.” (Ex. 1008 at 2:38-42).

274. Additionally, both Halperin and Arnold approach some aspects of this problem in similar ways. For example, Arnold discloses detecting “anomalies” that indicate malicious software. Arnold explains:

The first step in the process of this invention detects anomalous system behavior of a type that may indicate the presence of an undesirable informational state resulting from the presence of a virus or some other undesirable software entity, such as a worm or a Trojan Horse. The detection of anomalous behavior within a computer or computer network can be accomplished through the use of known techniques, preferably a technique that detects anomalies that may indicate a virus. (Ex. 1008 at 4:60-5:15).

275. A POSITA would have recognized that Halperin’s system for “detecting malicious activity in a computer network” as a “known technique... that detects anomalies...” (Ex. 1003 at Abstract, Ex. 1008 at 4:60-5:15). For example, the following events disclosed by Halperin would have been recognized by a POSITA as “anomalies:”

- a. Sending messages to more than a predefined number of users during a predefined time.

- b. Sending messages not as a result of a direct user interaction with the Graphic User Interface (GUI) of the message software, but rather as the result of a directive from a software application.
- c. Modifying operating system files such as the Microsoft Windows registry.
- d. Deleting more than a predefined number of files on the computer's hard disk during a predefined period of time.
- e. Loading a new software application into the computer's RAM.
- f. Sending a file attached to a message several times from the same user.
- g. Sending a file attachment of a specific type (e.g., .exe, .doc, zip).
- h. Attempting to contact previously unused or unknown IP addresses or IP sockets.

(Ex. 1003 at [0038]-[0046]).

276. Another similarity between Halperin and Arnold in performing reactive analysis is that both references use multiple samples in recognizing the new malware. Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.*** (Ex. 1003 at [0047], emphasis added).

277. This disclosure from Halperin expresses that the same behavior must be detected at two or more computers in order for the server to determine that a virus may have infected the machines.

278. Similarly, Arnold discloses using multiple samples of malware to create the signature. For viruses, Arnold explains that the modified file contains a copy of the virus that modified it and Arnold uses that to extract the virus data from multiple samples. “A comparison of each modified decoy program with its corresponding uninfected version enables a copy of the virus to be isolated from each decoy.” (Ex. 1008 at 7:5-8).

279. With respect to worms, Arnold explains that this kind of malware modified scripts that launch the worm. Thus, “even though worms do not necessarily rely upon modification of other executables, modification detection techniques may still be employed for detecting attempts by the worm to modify login or startup scripts to include invocations of itself.” (Ex. 1008 at 21:20-25). In this case, then, either the worm or the script may cause an event. The worm causes file changes to the scripts; but the scripts, when run, load the worm into memory.

280. Arnold teaches that a signature may be extracted by correlating data between modified files (such as the scripts or virus-infected files) or identical files (such as worms). In order to generate the signature, Arnold identifies common sequences between the files.

In order to maximize the likelihood that a chosen signature will appear in all or nearly all progeny of the virus, it is preferred to filter out all portions of the virus which are observed to or are likely to vary from one instance of the virus to another.
(Ex. 1008 at 7:11-15).

If all versions are identical, or if there is only one version, the entire virus is considered to be "invariant", and can be passed along to block B Otherwise, if the various isolated computer virus programs differ, but have substantial areas in common, it can be assumed that the virus alters itself when it replicates. As a result, ***only the common areas are marked for further examination by the heuristics employed in Block B of FIG. 4.*** This procedure is illustrated in FIG. 6a, in which the variant and "invariant" portions of a virus are identified by comparing three instances of it. In FIG. 6a, each byte is shown expressed in a hexadecimal representation, and an invariant portion is identified. The length of the invariant sequence of bytes is exemplary. In practice, invariant sequences of bytes are often

considerably longer than five bytes, and more than one such portion typically appears within the virus.
(Ex. 1008 at 7:25-49, emphasis added).

281. Once the “invariant” sections of the copies are identified, a “signature” is extracted. Here, however, Arnold provides a proactive component as well. As cited above, Arnold attempts to find the common patterns between viruses so that even if subsequent versions are slightly different, Arnold is proactively ready to deal with them. This proactive signature is added to the antivirus database.

Briefly stated, the technique disclosed in this commonly assigned U.S. Patent Application implements a signature scoring technique which processes a large corpus of typical programs, such as PC-DOS and OS/2 binary executables, collects statistics on n-gram frequencies, and uses this information to estimate the probability of finding a given sequence of bytes (with a specified number of allowed mismatches) in the programs of the corpus. Once the signatures have been extracted, they are added permanently to a database used by the virus scanner, and all relevant media are scanned and cleaned up as in Step B.
(Ex. 1008 at 9:49-60).

282. Notably, Halperin disclosed creating a very basic signature in the form of a “checksum.” In this example, Halperin identifies repeated malicious email attachments, computes a checksum, and transmits it to other computers.

In addition, when malicious software has been identified in several computing devices within a group, it is possible to isolate the mechanism that has been spreading the malicious software. For example, where malicious software is spread by e-mail, the e-mail attachment that when activated causes the malicious Software to spread may be identified. A characteristic code may be generated for the attachment that distinguishes it from other Such attachments. This may be done using well known “checksum” algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the checksum to identify suspicious malicious Software upon arrival at these computers.
(Ex. 1003 at [0073]).

283. But Halperin's checksum is not proactive. It cannot detect any changes in the malware. A POSITA would have recognized that Arnold provides a better mechanism, like Mankins, for proactively dealing with threats. At the same time, such an approach is entirely compatible with Halperin's malware identifying techniques.

284. Thus, a POSITA would have recognized that Halperin, Mankins, and Arnold are compatible.

285. A POSITA would have also been motivated to combine Arnold with Halperin and Mankins. As I discussed with respect to Claim 1, Halperin discloses a wide range of examples of events as well as numerous approaches to responding to those events. A POSITA would have been motivated to respond to those events if/when they are detected on their systems. However, A POSITA would have known that Halperin's responses do not cover all of the events identified therein. For example, Halperin disclosed that detected anomalous events could include an object "[l]oading a new software application into the computer's RAM." (Ex. 1003 at [0043]). However, Halperin does not discuss how to respond to this event. A POSITA, upon detection of an event identified by Halperin as anomalous, would be motivated to find another reference that teaches how to respond to such an event. Notably, Mankins' security mask also provides a general framework (i.e., identifying "normal" behavior during training and blocking "abnormal" behavior that appears later), but provides no details about how to identify "normal" or "abnormal" software loading. A POSITA would have been motivated to look for other references, such as Arnold, that teach how to recognize "good" loading over "bad" loading, and how to handle the event when it is determined to be "bad".

286. In particular, a POSITA would have been motivated to draw on Arnold's teaching for dealing with worms and identifying their spawning processes. Arnold explains that "the

method proceeds to Step B, in which that process, all parents or children of that process, and perhaps all other processes in memory, are scanned for known worms.” (Ex. 1008 at 21:30-42). Arnold further explains “... the worm propagates by **spawning processes** from one machine to another.” (Ex. 1008 at 21:10-12, emphasis added). A POSITA would have known that “spawning processes” or “invocations of [the worm]” are synonyms of “loading a new software application into RAM.”

287. Arnold’s analysis of worms provides guidance on what is a “good” launcher and a “bad” launcher. Arnold teaches observing launcher’s for malicious changes (e.g., caused by the worm) spread across multiple samples. These changes, typically caused by a worm inserting itself into the launcher, can be identified as a signature. Arnold discloses that, “...even though worms do not necessarily rely upon modification of other executables, modification detection techniques may still be employed for detecting attempts by the worm to modify login or startup scripts to include invocations of itself...” (Ex. 1008 at 21:18-29). Arnold also teaches that the worm itself can have a signature (which would enable a corrupted launcher to be identified as well). .

In order to maximize the likelihood that a chosen signature will appear in all or nearly all progeny of the virus, it is preferred to filter out all portions of the virus which are observed to or are likely to vary from one instance of the virus to another.
(Ex. 1008 at 7:11-15).

If all versions are identical, or if there is only one version, the entire virus is considered to be "invariant", and can be passed along to block B Otherwise, if the various isolated computer virus programs differ, but have substantial areas in common, it can be assumed that the virus alters itself when it replicates. As a result, ***only the common areas are marked for further examination by the heuristics employed in Block B of FIG. 4.*** This procedure is illustrated in FIG. 6a, in which the variant and "invariant" portions of a virus are identified by comparing three instances of it. In FIG. 6a, each byte is shown expressed in a hexadecimal representation, and an invariant portion is identified. The length of the invariant sequence of bytes is exemplary. In practice, invariant sequences of bytes are often

considerably longer than five bytes, and more than one such portion typically appears within the virus.
(Ex. 1008 at 7:25-49, emphasis added).

288. Once the “invariant” sections of the copies are identified, a “signature” is extracted.

Briefly stated, the technique disclosed in this commonly assigned U.S. Patent Application implements a signature scoring technique which processes a large corpus of typical programs, such as PC-DOS and OS/2 binary executables, collects statistics on n-gram frequencies, and uses this information to estimate the probability of finding a given sequence of bytes (with a specified number of allowed mismatches) in the programs of the corpus. Once the signatures have been extracted, they are added permanently to a database used by the virus scanner, and all relevant media are scanned and cleaned up as in Step B.
(Ex. 1008 at 9:49-60).

289. Thus, using signatures on the launcher or on the worm itself, a corrupted launcher can be identified.

290. It would have been obvious to a POSITA to use Arnold’s worm signature processing with Halperin and Arnold’s combined approach to securing a system. Using Arnold, Halperin can both expand reactive detection of worms as well as expand Mankins’ mask to proactively prohibit start-up scripts from launching worms. Arnold explains that when a worm is launched or spawned from another process, the correct remediation is to shut down all processes in the tree. Arnold explains:

The user is informed of suspicious activity and given a choice as to whether to continue running or to suspend the offending process (and/or the family tree of processes to which it is related) pending further investigation.

If the user chooses to suspend the process, the method proceeds to Step B, in which that process, all parents or children of that process, and perhaps all other processes in memory, are scanned for known worms. Cleanup involves killing active worm processes (determined by tracing the process tree), and deleting worm executables and auxiliary files from storage media.
(Ex. 1008 at 21:30-42).

291. A POSITA would have found this combination straight-forward and with a high likelihood of success. As described in more detail in the claims and limitations below, the steps inserted from Arnold are similar or analogous to steps already in Halperin and/or Mankins.

292. A POSITA would have recognized that this combination enables the detection of worm-like malware and enables more effective behavior limiting of monitored processes.

B. Claim 6 is rendered obvious by Halperin in view of Mankins in further view of Arnold

293. Claim 6 of the '250 patent, which depends from Claim 3, reads:

A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

294. I have already explained that Halperin in view of Mankins render obvious Claim 3. Halperin in view of Mankins in further view of Arnold discloses the additional requirements of Claim 6.

295. Halperin disclosed determining a relationship between parent objects and child objects (from a process standpoint). Specifically, Halperin disclosed a server tracking the “Loading a new software application into the computer’s RAM.” (Ex. 1003 at [0043]). A POSITA would have understood that when one process launches another process, the first is considered the “parent” process of the second.

296. Arnold disclosed a method of creating signatures for worms and tracing other scripts that spawned them. Arnold explains that worms modify scripts to subsequently (re)launch the worm. Thus, “even though worms do not necessarily rely upon modification of other executables, modification detection techniques may still be employed for detecting attempts by the worm to modify login or startup scripts to include invocations of itself.” (Ex. 1008 at 21:20-25). In this case, then, either the worm or the script may need to have a signature created that can be

sent to Halperin's central computer. The worm causes file changes to the scripts; but the scripts, when run, load the worm into memory (both of which would be relevant to Halperin's "Loading a new software application into a computer's RAM" event). Once this kind of anomaly is detected, Arnold teaches extracting a signature.

297. Arnold teaches that a signature may be extracted by correlating data between modified files (such as the scripts or virus-infected files) or identical files (such as worms). In order to generate the signature, Arnold identifies common sequences between the files.

In order to maximize the likelihood that a chosen signature will appear in all or nearly all progeny of the virus, it is preferred to filter out all portions of the virus which are observed to or are likely to vary from one instance of the virus to another.
(Ex. 1008 at 7:11-15).

If all versions are identical, or if there is only one version, the entire virus is considered to be "invariant", and can be passed along to block B
Otherwise, if the various isolated computer virus programs differ, but have substantial areas in common, it can be assumed that the virus alters itself when it replicates. As a result, ***only the common areas are marked for further examination by the heuristics employed in Block B of FIG. 4.*** This procedure is illustrated in FIG. 6a, in which the variant and "invariant" portions of a virus are identified by comparing three instances of it. In FIG. 6a, each byte is shown expressed in a hexadecimal representation, and an invariant portion is identified. The length of the invariant sequence of bytes is exemplary. In practice, invariant sequences of bytes are often considerably longer than five bytes, and more than one such portion typically appears within the virus.
(Ex. 1008 at 7:25-49, emphasis added).

298. Once the "invariant" sections of the copies are identified, a "signature" is extracted.

Briefly stated, the technique disclosed in this commonly assigned U.S. Patent Application implements a signature scoring technique which processes a large corpus of typical programs, such as PC-DOS and OS/2 binary executables, collects statistics on n-gram frequencies, and uses this information to estimate the probability of finding a given sequence of bytes (with a specified number of allowed mismatches) in the programs of the corpus. Once the signatures have been extracted, they are added permanently to a database used by the virus scanner, and all relevant media are scanned and cleaned up as in Step B.

(Ex. 1008 at 9:49-60).

299. Thus, Arnold teaches that each of these files used in creating the signature (copies of virus-infected files, worms, or modified scripts) A POSITA would have been motivated to combine Halperin and Arnold as explained in Section X.A. Specifically with reference to this limitation, a POSITA would have been motivated to combine these disclosures such that the identity of the object initiating an event is included in order to apply Arnold's method for extracting a signature and transmitting it to Halperin's central server.

300. A POSITA would have been further motivated to combine Arnold and Mankins. Specifically, and as described with respect to Claim 1, a POSITA would have already been motivated to combine Mankins with Halperin in order to create the proactive defense of Mankins' masking. Even elements determined to not be malware might need to be watched for subsequent deviation as already discussed.

301. Similarly, once Halperin, combined with Arnold, note potential modifications to a start-up script, it may not **yet** be known if these modifications are malicious. Or, an accurate worm signature may not yet be known as Arnold discloses needing multiple samples. A POSITA would have been motivated to combine Arnold's analysis of scripts that launch worms into Mankins' behavioral mask. In the combined system, Mankins' mask would monitor the start-up scripts for the software they load into memory. This data, monitored at Halperin's central server, can be used to detect if a worm is being launched (once a signature for the worm is created).

302. As explained above, Arnold discloses that, "...even though worms do not necessarily rely upon modification of other executables, modification detection techniques may still be employed for detecting attempts by the worm to modify login or startup scripts to include invocations of itself..." (Ex. 1008 at 21:18-29).

303. When worms behave as described by Arnold to modify scripts, the scripts become part of the worm's operations. Accordingly, Arnold explains that "the method proceeds to Step B, in which that process, all parents or children of that process, and perhaps all other processes in memory, are scanned for known worms." (Ex. 1008 at 21:30-42). Arnold further explains "... the worm propagates by spawning processes from one machine to another." (Ex. 1008 at 21:10-12, emphasis added).

304. When the program is running, its behavior is compared to the bloom filter. If the program attempts to deviate from the "trained" behavior, the system raises an alert.

To define normal behavior, a particular computer program 130 may be invoked and operated ***under training conditions*** in order to determine a set of operations initiated by the computer program 160.
(Ex. 1004 at 6:63-66, emphasis added).

The build subsystem 160 is configured to ***store*** the generated profiles of normal program behavior (e.g., sets of unique system-call sequences) compactly by using ***[a] Bloom filter*** 175.
(Ex. 1004 at 7:22-25, emphasis added).

The build subsystem 160 may be configured to associate a generated Bloom filter data object 180 to its corresponding computer program 130. In particular, the build subsystem 160 may cause the Bloom filter data object 180 to be ***appended to the executable image*** of the computer program 130 in a manner that will cause the Bloom filter data object 180 to be automatically embedded in the OS kernel 120 upon invocation of the computer program 130.
(Ex. 1004 at 7:59-66, emphasis added).

The intrusion detection system 100 may be configured to ***automatically*** generate Bloom filter data objects 180 for computer programs 130 and then ***automatically*** use the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time.
(Ex. 1004 at 8:8-13, emphasis added).

305. Moreover, A POSITA would also have noted that Mankins explicitly promotes a "distributed" environment as an option: "...the intrusion detection system 100 may be

implemented in a distributed fashion over a network of computing devices.” (Ex. 1004 at 4: 29-31). As discussed with respect to limitation 1[f], in this distributed environment, Mankins monitoring of operations can include monitoring at Halperin’s central server. Detection of operations that launch worms can be enhanced as updated worm definitions are calculated.

306. A POSITA would have recognized that such a combination enables start-up scripts to continue operating while being carefully monitored for aberrant behavior collected from across the distributed network. As is promoted by Halperin, once there is sufficient data, the behavior can be recognized as undesirable. Combined with Mankins, it can also be blocked or disabled. A POSITA would have been motivated to perform this combination in order to maximize the stated goals of Halperin and Mankins at the very least with respect to the operation of “loading a software application into memory.”

307. Thus, I conclude that Halperin in view of Mankins in view of Arnold discloses the additional requirements of Claim 6.

C. Claim 18 is rendered obvious by Halperin in view of Mankins in further view of Arnold

308. Claim 18 of the ’250 patent, which depends from Claim 15, reads:

Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

309. See analysis of Claim 6.

310. Thus, I conclude that Halperin in view of Mankins in view of Arnold discloses Claim 18.

XI. GROUND 3: CLAIMS 7-11, 19-23, AND 25-30 ARE RENDERED OBVIOUS BY HALPERIN IN VIEW OF MANKINS IN FURTHER VIEW OF KIRBY IN FURTHER VIEW OF CARPENTIER

311. As explained in the sections and paragraphs below, Halperin in view of Mankins in further view of Kirby in further review of Carpentier renders obvious the identified claims.

A. Motivation to Combine Halperin, Mankins, Kirby, and Carpentier

312. A POSITA would have been motivated to combine Halperin, Mankins, Kirby, and Carpentier. I have already explained in Section IX.A. that it would have been obvious to combine Halperin and Mankins. I incorporate this analysis by reference.

313. A POSITA would have been further motivated to combine Halperin and Mankins with Kirby. Halperin already disclosed using “checksums” for identifying malware. “A characteristic code may be generated for the attachment that distinguishes it from other Such attachments. This may be done using well known ‘checksum’ algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the checksum to identify suspicious malicious software upon arrival at these computers.” (Ex. 1003 at [0073]). A POSITA would have known that hashing is one method of producing a checksum.

314. Kirby, which is also directed to securing systems against malicious software, explicitly uses the well-known method of hashing to identify signatures on the malicious data. With respect to generating an identifier of the malware, Kirby teaches:

A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator accumulates successive realtime signatures in this manner for comparison with subsequent incoming transmissions.
(Ex. 1006 at 4:10-16).

315. The hashes, Kirby discloses, build up a set of signatures for subsequent comparison. “A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator therefore accumulates a baseline of realtime signatures for comparison with subsequent incoming transmissions.” (Ex. 1006 at 7:41-62). A POSITA would have understood that using a hash algorithm is an extremely efficient method for comparing two objects. As Kirby says, “The security agent 110 performs a check to determine if the computed signature is indicative of undesirable activity, as shown at step 318. Typically, such a comparison involves computing a hash on the incoming object, and comparing the hash to the previously generated realtime signatures 176A in the repository. ***As indicated above, the realtime hash in response to passing the behavior check is an efficient comparison mechanism to determine if the incoming object has previously been associated with an undesirable transaction based on the behavior check at step 307.***” (Ex. 1006 at 13:43-62).

316. Carpentier also discloses using hashing for identifying content. Carpentier explains that, by using hashing, one can identify content by the content itself, rather than by changing labels or names. Carpentier explains that a “...cryptographic hash function used to generate the asset content identifier can be applied to the contents of the asset (or to information which is believed to be the original asset) at any time and the match between the asset and the asset content identifier can then be verified.” (Ex. 1013 at [0034]). The content identifier is used to identify rather than “file name, date, location, or any other such label which may be changed independently of the asset contents.” (*Id.*). Using a content identifier, rather than an arbitrary label, “an asset may be obtained from any location and the integrity of the contents may be reliably verified by

recomputing the asset hash and comparing it to a stored asset content identifier...” (Ex. 1013 at [0013]).

317. In addition to a hash over the contents of the file itself, Carpentier also discloses using a cryptographic hash on metadata about the file as stored in a “descriptor file.”

An example of a descriptor file is shown in FIG. 5. The other asset information included with the descriptor file may include directory information about where the files are located, file names, file sizes, time and date stamps for each file included, ownership of such files, and other meta data, as is described below. The descriptor file may contain similar descriptions of binary assets which were not originally files and which may not be used to create unique files. Such assets represent binary sequences which are records, elements, or clips of larger sequences or collections. These assets are treated in a manner similar to files and are returned to their user for processing through appropriately designed binary interfaces. (Ex. 1013 at [0031]).

318. Fig. 5, referenced above, is reproduced hereinbelow:

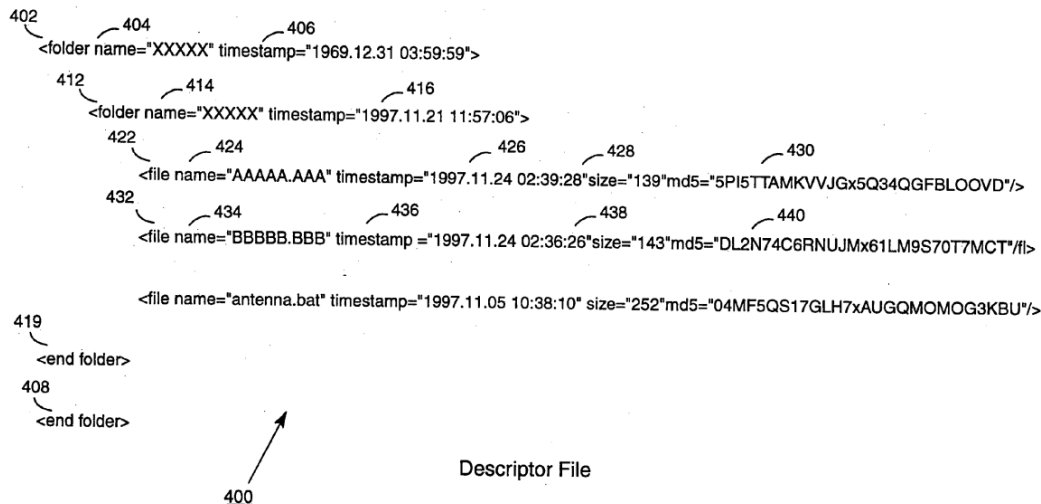


FIG. 5

319. Carpentier explains that the descriptor file is also hashed: “[a] cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed.” (Ex. 1013 at [0033]). The descriptor file is useful in identifying metadata about files, such as the file path

(folder structure) location, size, and timestamps as shown in Fig. 5 above. The hash of the descriptor file is an identifier for the content. “A cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed which, in one embodiment, becomes the e-CLIP by which the collected files may be referenced, found, and verified.” (Ex. 1013 at [0033]).

320. Thus, Carpentier does not just hash the data of files, but also the metadata such as the file path, name, size, and other information referenced in Fig. 5.

321. A POSITA would have recognized that methods of hashing data and metadata, as disclosed by both Kirby and Carpentier, would fit naturally with the data transmissions required by Halperin (including checksum transmissions). A POSITA would have found it obvious to apply Kirby’s teaching in order to build up a “baseline” set of signatures for subsequent comparison, as explained in the citations above (e.g., Ex. 1006 at 3:35-58) and especially in order for the efficient comparison capabilities Kirby describes, as discussed above (e.g., Ex. 1006 at 13:43-62). A POSITA would have known that sending a hash fingerprint for comparison over a network is more efficient than sending the whole content. Moreover, Halperin fits naturally with the malware identification tracking of Kirby because Halperin discloses a centralized server that receives data about a virus from multiple computers. Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.*** (Ex. 1003 at [0047], emphasis added).

322. Although Halperin discusses the identification in terms of behavior, Halperin discloses that the behavior is associated with specific objects. This allows it to, once suspicious

objects are identified, transmit data about the object to other computers. “A characteristic code may be generated for the attachment that distinguishes it from other such attachments. This may be done using well known ‘checksum’ algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the checksum to identify suspicious malicious software upon arrival at these computers.” (Ex. 1003 at [0073]). A POSITA would have recognized that a database of detected signatures, as disclosed by Kirby, would provide the functionality that Halperin describes. Halperin also uses checksums to avoid duplicates and a POSITA would have recognized that a database of detected signatures also eliminates duplicates because each piece of content appears in the database once based on its hash (see also Ex. 1006 at 12:40-45 and 13:34-61 noting these deduplication advantages for its method).

323. A POSITA would also recognize that hashing produces a unique fingerprint for efficient comparison and deduplication. A POSITA would have known that this is why Kirby describes the hash comparison operation as “efficient.” A POSITA would have known that a hash function produces a binary sequence fingerprint that is (effectively) a unique identifier. This means that no “collisions” are expected (where two files have the same fingerprint). As Carpentier explains further,

The cryptographic hash function used to generate the asset content identifier can be applied to the contents of the asset (or to information which is believed to be the original asset) at any time and the match between the asset and the asset content identifier can then be verified. The integrity of this approach is based on the strength of the cryptographic hash functions used to create the asset content identifiers. ***The MD5 algorithm, for example, which is used in a preferred embodiment can be demonstrated to virtually guarantee a unique asset content identifier for any binary sequence, no matter how similar or different it may be to/from another sequence.*** As such, two assets whose MD5 digests are the same can be assumed to have identical binary sequences (contents) and two assets whose MD5 digests are different can be assumed to have different binary sequences.

(Ex. 1013 at [0034], emphasis added).

324. As Carpentier explains, this permits the hash to be used as a unique identifier for content. Additionally, Carpentier's descriptor file, described above, enables the identification of metadata about the file. As shown in Figure 5 (above), this descriptor file contains filenames, timestamps, and other data about content (Ex. 1013 at [0031]). The descriptor file is then hashed and the hash forms the identifier for the descriptor file and, in turn, an identifier for the contents (including the file and file metadata) that were described. This hash of the descriptor file enables a hash (or checksum) to identify not only the content, but the specific location of a file in a file system, its size, and timestamps.. (Ex. 1013 at [0060]).

325. A POSITA would have found it obvious to transmit Carpentier's content-identifier fingerprints (hashes) as identifiers in the combined system of Halperin-Mankins-Kirby. That is, the hash/checksum transmission system and processes of Halperin and Kirby (combined above) would be obvious to a POSITA to use for the identifiers of Carpentier in order to efficiently identify file metadata information, such as location information, in addition to the file content. Additionally, a POSITA would have recognized the advantages of using hashes of the file's description or metadata to identify the file in addition to the contents as disclosed in Carpentier and been motivated to modify the Halperin-Mankins-Kirby hashing transmission system for file identification to hash metadata of the files on the network. This modification would have been valuable to Halperin, which discloses communicating information about files, such as their names and locations, distributed across a network to a central location. For example, Halperin identifies files in specific locations (such as "system" files): "...target behavior includes... Modifying operating system files..." (Halperin at [0038] and [0041]). Halperin also identifies files with a specific name: "A certain percentage of computers in the network deleting a file by the *same name*

within the last 1 hour.” (Halperin at [0051], emphasis added). Notably, the timing requirements Halperin discloses (e.g., “within the last 1 hour”) would also be supported by hash/checksums that include timestamps. Thus, a POSITA would have found it obvious to incorporate Carpentier’s enhanced use of hashing into Halperin’s operations.

326. Moreover, because of the ease of compatibility, the overlapping domains, and the use of standard functionalities, a POSITA would have had a strong expectation of success in combining these four references.

B. Claim 7 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

327. Claim 7 of the ’250 patent, which depends from Claim 1, reads:

A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.

328. I have already explained my analysis that Halperin in view of Mankins render obvious Claim 1 in Section X.B. I incorporate my analysis of this Claim, from which Claim 7 depends, into this section by reference. As already explained within that section, Halperin sends data about behavior from remote computers to the central computer.

329. Moreover, Halperin discloses the use of checksums. A POSITA would have understood that hashing is a form of checksum. Halperin discloses identifying malware (e.g., malicious email attachments) with a checksum to differentiate it from other malware. Moreover, these codes can be shared between computers on the network. “A characteristic code may be generated for the attachment that distinguishes it from other such attachments. This may be done using well known ‘checksum’ algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the

checksum to identify suspicious malicious software upon arrival at these computers.” (Ex. 1003 at [0073]). Halperin’s computers use the checksums to identify duplicates.

330. To the extent that Halperin does not explicitly disclose the use of “hashing” as a checksum, it would have been obvious in view of Kirby. Kirby uses hashing, a known form of checksum, to build a database of known signatures.:

A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator accumulates successive realtime signatures in this manner for comparison with subsequent incoming transmissions.
(Ex. 1006 at 4:10-16).

331. The hashes, Kirby discloses, build up a set of signatures for subsequent comparison. “A signature generator, such as a hash engine, then computes a realtime signature, such as a hash, on the suspect object. The realtime signature is written to a signature database for comparison with future suspect objects in incoming transmissions. The signature generator therefore accumulates a baseline of realtime signatures for comparison with subsequent incoming transmissions.” (Ex. 1006 at 7:41-62).

332. To the extent that Halperin in view of Kirby does not disclose sending data in the form of a key, it would have been obvious in view of Carpentier. Carpentier also discloses using hashing for identifying content. Carpentier explains that, using hashing, one can identify content by the content itself, rather than by changing labels or names. Carpentier discloses that a “...cryptographic hash function used to generate the asset content identifier can be applied to the contents of the asset (or to information which is believed to be the original asset) at any time and the match between the asset and the asset content identifier can then be verified.” (Ex. 1013 at [0034]). The content identifier is used to identify rather than “file name, date, location, or any

other such label which may be changed independently of the asset contents.” (*Id.*). Using a content identifier, rather than an arbitrary label, “an asset may be obtained from any location and the integrity of the contents may be reliably verified by recomputing the asset hash and comparing it to a stored asset content identifier...” (Ex. 1013 at [0013]).

333. A POSITA would have been motivated to use the hash of the malicious content to communicate about data between servers. As Carpentier explains, this identification system enables identifying content without worrying about file name, date, location, or...” other labels that might change, especially between computer systems.

334. In addition to a hash over the contents of the file itself, Carpentier also discloses using a cryptographic hash on metadata about the file as stored in a “descriptor file.”

An example of a descriptor file is shown in FIG. 5. The other asset information included with the descriptor file may include directory information about where the files are located, file names, file sizes, time and date stamps for each file included, ownership of such files, and other meta data, as is described below. The descriptor file may contain similar descriptions of binary assets which were not originally files and which may not be used to create unique files. Such assets represent binary sequences which are records, elements, or clips of larger sequences or collections. These assets are treated in a manner similar to files and are returned to their user for processing through appropriately designed binary interfaces. (Ex. 1013 at [0031]).

335. Fig. 5, referenced above, is reproduced hereinbelow:

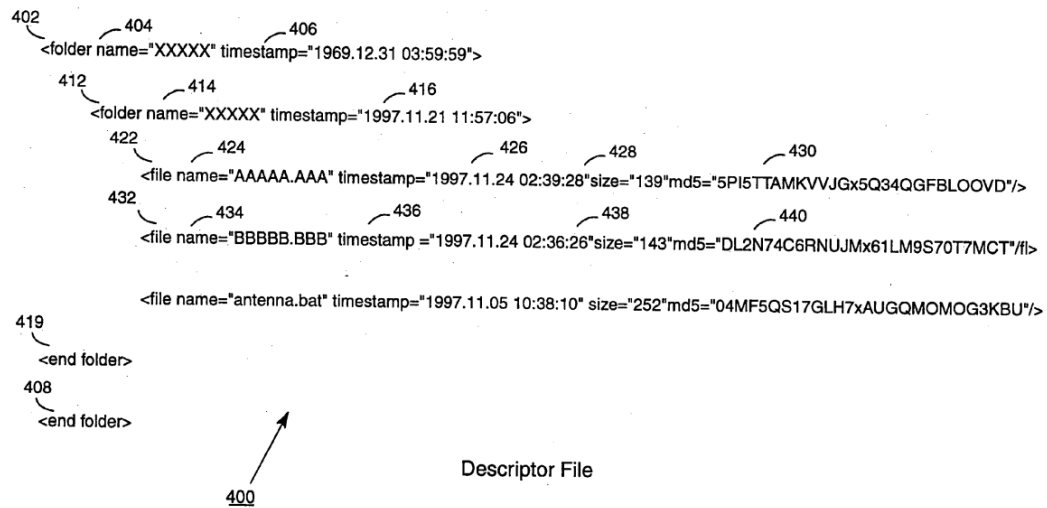


FIG. 5

336. Carpentier explains that the descriptor file is also hashed: “[a] cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed.” (Ex. 1013 at [0033]). The descriptor file is useful in identifying multiple content files (e.g., in a list) and the hash representing a pointer to the set of files. “Once the asset content identifier for each of the selected assets is generated, it is included in an asset list along with other relevant information in a descriptor file.” (Ex. 1013 at [0031]). “A cryptographic hash file identifier (asset content identifier) for the stored descriptor file is then computed which, in one embodiment, becomes the e-CLIP by which the collected files may be referenced, found, and verified.” (Ex. 1013 at [0033])

337. A POSITA would have found it obvious to transmit data about suspicious programs from the remote computer to the central computer using hashing. Halperin already discloses transmitting checksums to identify files. Kirby discloses using hashing (a form of checksumming) to track malicious files and Carpentier further discloses transmitting hashes as identifiers including hashes over metadata for including aggregations of assets.

338. A POSITA would have been motivated to combine Halperin and Mankins with Kirby and Carpentier. I incorporate my analysis from Section XI.A. into this section by reference.

339. Thus, I conclude that Halperin in view of Mankins in further view of Kirby in further view of Carpentier discloses Claim 7.

C. Claim 8 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

340. Claim 8 of the '250 patent, which depends from Claim 7, reads:

A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

341. I have already explained with reference to Claim 7 that Halperin in view of Mankins in further view of Kirby and Carpentier discloses transmitting hashed data from remote computers to the central computer for comparison. I incorporate my analysis of this claim into this section by reference.

342. As I explained with reference to that claim, Kirby and Carpentier both disclose hashing an entire file and a POSITA would have found it obvious to combine Carpentier with Halperin and Kirby. In this combination, a POSITA would have transmitted the Halperin data using hashing to produce a unique fingerprint that is used for comparison at the central computer. A POSITA would have known that when executable files (such as malware) are hashed, the hash output represents the entire file which, by definition, includes all the executable instructions therein. Or, in other words, the executable instructions are constituted by the object and reflected in the output hash.

343. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 8.

D. Claim 9 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

344. Claim 9 of the '250 patent, which depends from Claim 7, reads:

A method according to claim 7, wherein the key has at least one component that represents data about said object.

345. I have already explained with reference to Claim 7 that Halperin in view of Mankins in further view of Kirby and Carpentier discloses transmitting hashed data from remote computers to the central computer for comparison. I incorporate my analysis of this claim into this section by reference.

346. As I explained with reference to that claim, Carpentier discloses hashing metadata about the file and a POSITA would have found it obvious to combine Carpentier with Halperin and Kirby. In this combination, a POSITA would have transmitted the Halperin data using hashing to produce a hash of a descriptor file for combinations of files, including the filename and timestamps of the data. As explained above, this metadata is data about the file, such as file name, file size, folder location, last time modified, and so forth. For illustration, I have reproduced Fig. 5 of Carpentier hereinbelow:

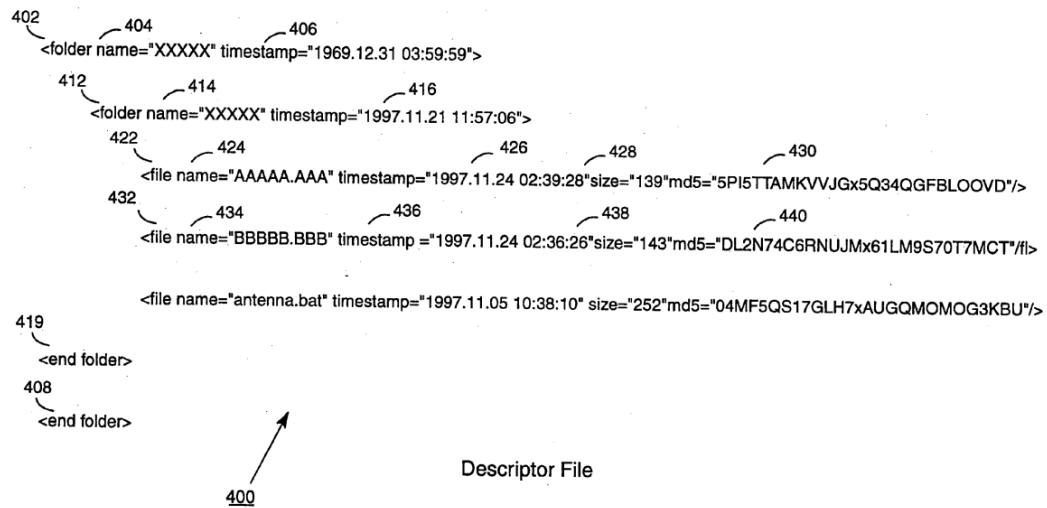


FIG. 5

347. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 9.

E. Claim 10 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

348. Claim 10 of the '250 patent, which depends from Claim 9, reads:

A method according to claim 9, wherein said data about said object includes at least one of the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

349. I have already explained with reference to Claim 7 that Halperin in view of Mankins in further view of Kirby and Carpentier discloses transmitting hashed data from remote computers

to the central computer for comparison. I further explained with reference to Claim 9 that Halperin in view of Mankins in further view of Kirby and Carpentier discloses hashing metadata and transmitting that data to the server for comparison. I include my analysis of these claims into this section by reference.

350. As I explained with reference to Claim 9, Carpentier discloses specifically using a timestamp of the file. A POSITA would have understood a timestamp associated with a file to include timestamps such as creation time and last-modified time. To the extent this claim requires both a creation and modification time, A POSITA would have found it obvious to include timestamps associated for both the file creation time and a last-modified time, in order to track the age of the file as well as how recently it has been updated. Also, a POSITA would have known that a just-created file would have a creation time and modification time that are the same.. Carpentier also includes filenames and folder locations. For illustration, I have reproduced Fig. 5 of Carpentier hereinbelow:

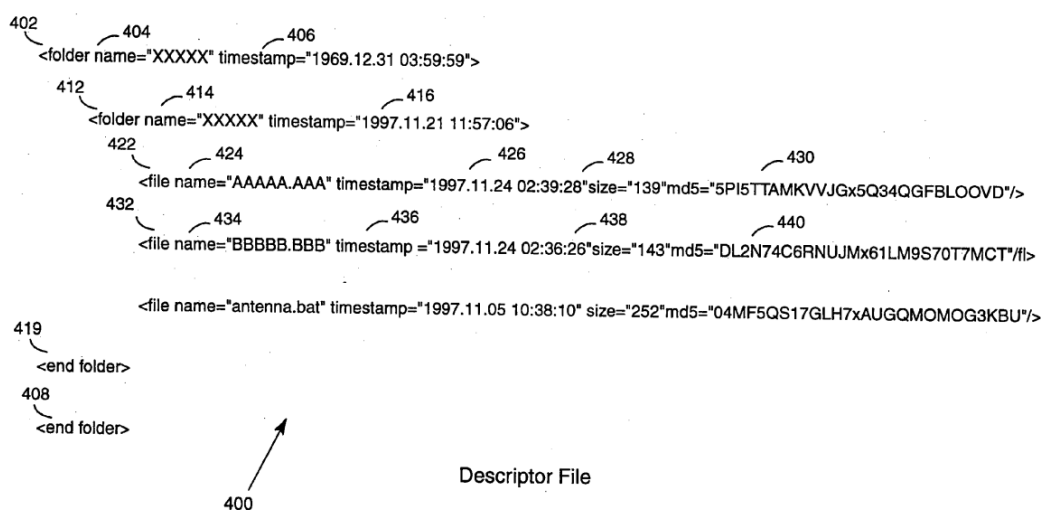


FIG. 5

351. As I explained with reference to Claims 7 and 9, Carpentier discloses hashing metadata about the file by hashing the descriptor file depicted in Fig. 5. Thus, a POSITA would have recognized that hashing Carpentier's descriptor file would have included folder name (e.g., folder names 404 and 414), file name (e.g., file names 424 and 434), file size (e.g., file sizes 428 and 438), and other information about the file such as its individual content hash (e.g., hashes 430 and 440). A POSITA would have found it obvious to combine Carpentier with Halperin and Kirby. In this combination, a POSITA would have transmitted the Halperin data using hashing to produce a short, fixed-size code (e.g., of the metadata) that is used for comparison at the central computer. As explained above, this metadata is data about the file, such as file name, file size, folder location, last time modified, and so forth. Moreover, a POSITA would have recognized that Carpentier's descriptor file hash identifies multiple files, enabling an identifier for more complex malware, such as malware spread across multiple files.

352. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses this claim.

F. Claim 11 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

353. Claim 11 of the '250 patent, which depends from Claim 7, reads:

A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

354. I have already explained with reference to Claim 7 that Halperin in view of Mankins in further view of Kirby and Carpentier discloses transmitting hashed data from remote computers to the central computer for comparison.

355. As I explained with reference to that claim, Carpentier discloses hashing the size of the file (through the descriptor file) and a POSITA would have found it obvious to combine

Carpentier with Halperin and Kirby. In this combination, a POSITA would have transmitted the Halperin data using hashing to produce a short, fixed-size code (e.g., of the metadata) that is used for comparison at the central computer.

356. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 11.

G. Claim 19 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

357. Claim 19 of the '250 patent, which depends from Claim 13, reads:

Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.

358. I have already explained my analysis that Halperin in view of Mankins render obvious Claim 13 in Section IX.I. I incorporate my analysis of this Claim, from which Claim 19 depends, into this section by reference. As already explained within that section, Halperin sends data about behavior from remote computers to the central server computer.

359. See analysis of Claim 7.

360. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 19.

H. Claim 20 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

361. Claim 20 of the '250 patent, which depends from Claim 19, reads:

Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

362. See analysis of Claim 8.

363. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 20.

I. Claim 21 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

364. Claim 21 of the '250 patent, which depends from Claim 19, reads:

Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

365. See analysis of Claim 9.

366. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 21.

J. Claim 22 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

367. Claim 22 of the '250 patent, which depends from Claim 21, reads:

Apparatus according to claim 21, wherein said data about said object includes at least one of the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

368. See analysis of Claim 10.

369. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 22.

K. Claim 23 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

370. Claim 23 of the '250 patent, which depends from Claim 19, reads:

Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

371. See analysis of Claim 11.

372. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 23.

L. Claim 25 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

373. Halperin in view of Mankins in further view of Kirby and Carpentier discloses all of the limitations of Claim 25.

1. Limitation 25[a]

374. Limitation 25[a] of the '250 patent reads:

A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

375. See analysis of limitation 1[a]. As I explained with reference to Claim 1, Halperin discloses a centralized server that functions as a “base computer” performing the operations recited in Claim 13 and its dependent claims. The centralized server is thus “constructed and configured” to perform the operations of these claims.

376. Thus, I conclude that Halperin discloses limitation 25[a].

2. Limitation 25[b]

377. Limitation 25[b] of the '250 patent reads:

providing from a remote computer to a base computer data about a computer object that is stored on the remote computer:

378. See analysis of limitation 1[b].

379. Thus, I conclude that Halperin discloses limitation 25[b].

3. Limitation 25[c]

380. Limitation 25[c] of the '250 patent reads:

the data including information about the behaviour of the object running on one or more remote computers and including one or more of executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

381. See analysis of limitation 1[b] and Claim 2.

382. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[c].

4. Limitation 25[d]

383. Limitation 25[d] of the '250 patent reads:

the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer;

384. See analysis of Claim 7.

385. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses limitation 25[d].

5. Limitation 25[e]

386. Limitation 25[e] of the '250 patent reads:

automatically generating a mask for the computer object, the mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data;

387. See analysis of limitation 1[f].

388. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[e].

6. Limitation 25[f]

389. Limitation 25[f] of the '250 patent reads:

executing the computer object at the remote computer;

390. See analysis of limitation 1[g].

391. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[f].

7. Limitation 25[f]

392. Limitation 25[f] of the '250 patent reads:

automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;

393. See analysis of limitations 1[h]-1[j].

394. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[f].

8. Limitation 25[h]

395. Limitation 25[h] of the '250 patent reads:

allowing the object to continue to run when behaviour of the computer object is permitted by the mask;

396. See analysis of limitation 1[i].

397. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[h].

9. Limitation 25[i]

398. Limitation 25[i] of the '250 patent reads:

disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and

399. See analysis of limitation 1[j].

400. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[i].

10. Limitation 25[j]

401. Limitation 25[j] of the '250 patent reads:

reclassifying the object as malware when the actual monitored behaviour extends beyond that permitted by the mask, wherein the base computer is arranged to receive information about the behaviour of the object running on one or more remote computers, wherein the step of generating a mask for that object comprises building the mask with the base computer in accordance with normal behaviour of the object determined from said received information

402. See analysis of limitation 1[k], limitation 1[b], and limitation 1[f].

403. Thus, I conclude that Halperin in view of Mankins discloses limitation 25[j].

404. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 25.

M. Claim 26 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

405. Claim 26 of the '250 patent, which depends from Claim 25, reads:

A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

406. See analysis of Claim 8.

407. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 26.

N. Claim 27 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

408. Claim 27 of the '250 patent, which depends from Claim 25, reads:

A method according to claim 25, wherein the key has at least one component that represents data about said object.

409. See analysis of Claim 9.

410. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 27.

O. Claim 28 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

411. Claim 28 of the '250 patent, which depends from Claim 27, reads:

A method according to claim 27, wherein said data about said object includes at least one of the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events

initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

412. See analysis of Claim 10.

413. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 28.

P. Claim 29 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

414. Claim 29 of the '250 patent, which depends from Claim 25, reads:

A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

415. See analysis of Claim 11.

416. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 29.

Q. Claim 30 is rendered obvious by Halperin in view of Mankins in further view of Kirby in further view of Carpentier

417. Claim 30 of the '250 patent, which depends from Claim 25, reads:

A non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

418. Halperin discloses that “the methods and apparatus disclosed herein may... have been described with reference to... hardware or software [and] it is appreciated that the methods and apparatus described herein may be readily implemented in hardware or software using conventional techniques.” (Ex. 1003 at [0092]). A POSITA would have known that software is stored, under almost every circumstance, in a non-transitory storage medium, such as hard-drives. A POSITA would have further known that software is equivalent to a “computer program” and that computer programs, by definition, contain “program instructions.” Thus, a POSITA would

have understood Halperin to disclose the software being stored in a non-transitory storage medium and containing program instructions for direction the operations of the system.

419. Thus, I conclude that Halperin in view of Mankins in further view of Kirby and Carpentier discloses Claim 30.

XII. GROUND 4: CLAIMS 1-5, 12-17, AND 24 ARE RENDERED OBVIOUS BY HALPERIN IN VIEW OF INOUE

420. As explained in the sections and paragraphs below, Halperin in view of Inoue renders obvious the identified claims.

A. Motivation to Combine Halperin and Inoue

421. A POSITA would have been motivated to combine Halperin and Inoue. Both Halperin and Inoue are both directed to the same problem. Specifically, both references attempt to protect computers on a network from intrusion. Halperin's approach is reactive insofar as it detects bad behavior and then acts to "contain" the infection. Inoue's approach is proactive insofar as it trains a guard system for acceptable behavior for a piece of software. The guard system then enforces that the software sticks to the trained behavior.

422. Nevertheless, despite the differences, a POSITA would have recognized the compatibility of these approaches. In particular, malware detection often includes both proactive and reactive systems. Because no one approach solves all malware problems, it is common to combine approaches to better obtain "defense in depth."

423. It would have been obvious to the POSITA that Inoue supports Halperin's goals. For example, Halperin discloses a system that learns as it goes, collecting data about "normal" and "abnormal" operations. Halperin, for example, discloses using increasing suspicious activity to eventually activate containment protocols.

In an alternative implementation of the buffer described above that is designed to reduce false alarms, should the server receive an invalid decoy

message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified. During the increased delay period, ***should additional suspicious messages be received, or should other suspicious behavior be detected, if the user and/or system administrator who is authorized to do so has not indicated that the activity is not virus related, only then does the server perform one or more virus containment actions.*** If, however, during the increased delay period no other suspicious activity is detected, or if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.
(Ex. 1003 at [0059], emphasis added.)

424. Nevertheless, Halperin does identify that one of the problems of this approach is that there must be a trade off between false alarms (detecting a “good” program as a bad one) and not detecting malware at all. As Halperin explains, “... the [probability] result is compared to a predefined threshold. If the calculated value is lower than the predefined threshold, the system can activate a virus alarm and possibly initiate virus containment actions. ***The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.***” (Ex. 1003 at [0090], emphasis added).

425. Inoue offers another alternative to a simple false-alarm/less-detection trade-off. A program believed to be benign can have a “sandbox” wrapped around it limiting the software to its “good” behavior. That way, if the software “gets worse,” it is stopped dynamically. As Inoue explains, “We show our implementation in a Java Virtual Machine is both effective and efficient at stopping a backdoor and a virus, and ***has a low false positive rate.***” (Ex. 1005 at p. 1, emphasis added.)

426. In other words, Inoue can be used as an active quarantine system. Because Halperin recognizes that suspicious activity can be a “false positive,” it would have been obvious to wrap

the suspect software into a system that can continue to monitor it along with a guard of acceptable behavior.

427. A POSITA would have been motivated to add these protections to Halperin, preventing damage when malware is not detected and improving the false positive rates. Moreover, because of the ease of compatibility, the overlapping domains, and the use of standard functionalities, a POSITA would have had a strong expectation of success in combining these two references.

B. Claim 1 is rendered obvious by Halperin in view of Inoue

428. Each of the limitations of Claim 1 of the '250 patent is rendered obvious by Halperin in view of Inoue as described below.

1. Limitation 1[a]

429. Limitation 1[a] of the '250 patent reads:

A method of classifying a computer object as malware, the method comprising:

430. 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 '250 is considered limiting, Halperin discloses a method of classifying a computer object as malware.

431. For example, Halperin discloses, “[a] method for detecting malicious activity in a computer network...” (Ex. 1003 at Abstract). Halperin goes on to explain that it detects malicious activity caused by viruses. In Figure 2, Halperin discloses containing a virus as shown below.

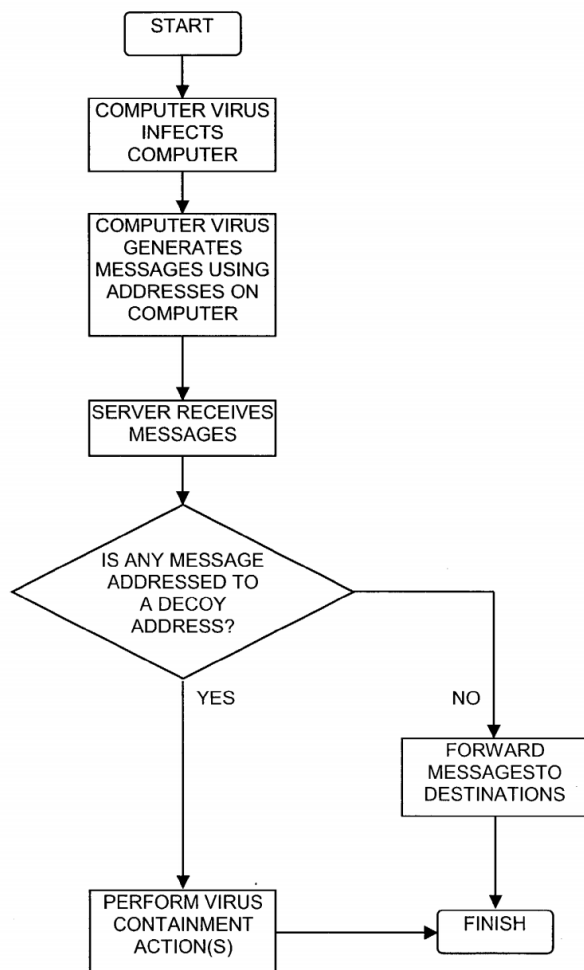


Fig. 2

432. Halperin also disclosed detecting a virus and transmitting information about the virus as shown in Figure 7, reproduced hereinbelow.

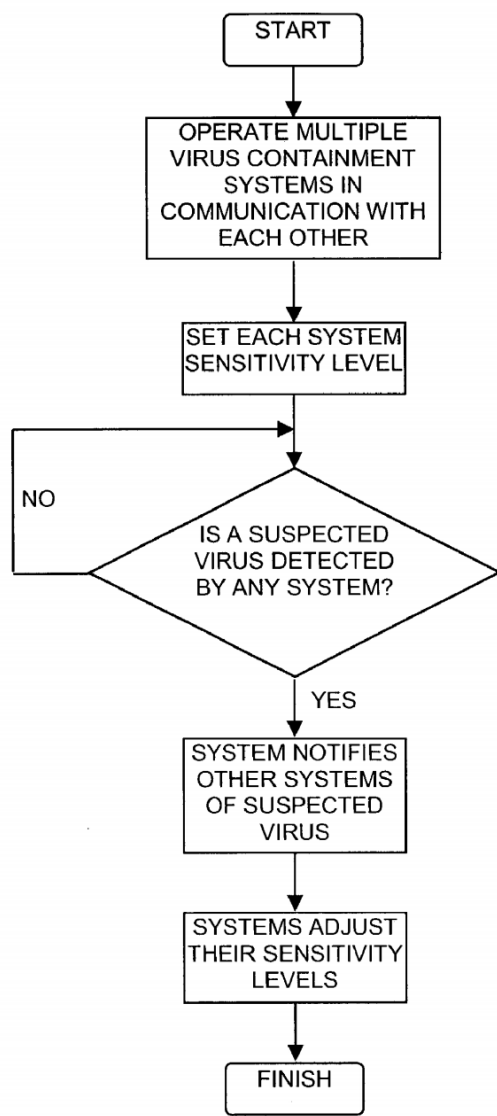


Fig. 7

433. Halperin also identifies viruses using behavior patterns as shown in Figure 11, reproduced hereinbelow.

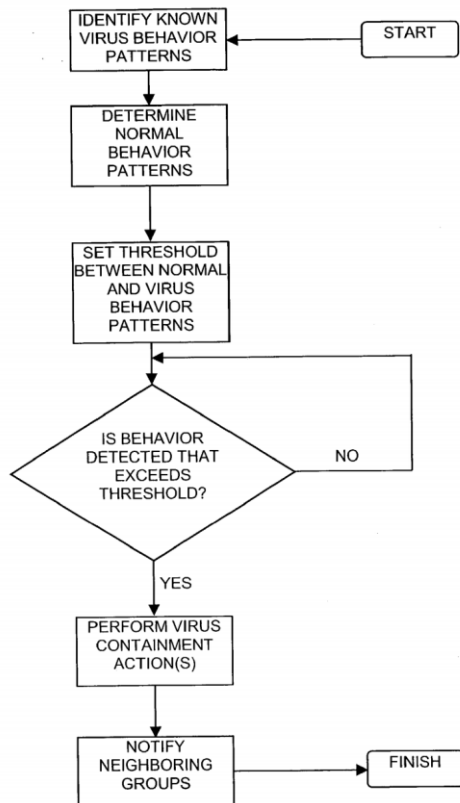


Fig. 11

434. A POSITA would have known that a virus is an example of malware.

435. Halperin also explains, “[t]he present invention relates to computer and computer network Security in general, and more particularly to detection and prevention of malicious computer programs.” (Ex. 1003 at [0002]). A POSITA would have known that “malicious computer programs” are malware.

436. In more detail, Halperin goes on to explain how the system works to identify malicious code (malware).

The proposed system works in two phases to *identify malicious code* (from here on referred to as “viruses”). **The two phases are: learning and detecting.** In the learning phase all the sensors are fully operational in the same network that is to be protected by the virus detection system. Alternatively it is possible, but less preferable, to do the learning phases and detecting phases on different computer networks. In such a case some kind of normalization processes may be required in porting the data from one network to another. In the learning phase all events are logged in a learning database similar to the database described above. This is done

for a substantial period of time, for example, one week. In addition, the system gets retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system. Generally, commercial antivirus software can quite effectively identify an existence of a virus in the system after it has received the relevant updates from the antivirus vendor, usually within a few days of the initial virus outbreak. This input is logged in a virus outbreak database. The time period in which it is estimated that the virus was active and the computers that were affected are logged in the virus outbreak database. ***In some cases the existence of a virus is identified only a few days after the virus started its activity.*** For that reason, the learning database is analyzed and put into use only after a waiting period (for example, two weeks). It is assumed that there is a high probability that most of the viruses in the system will be identified within the waiting period. (Ex. 1003 at [0087], emphasis added).

437. Thus, I conclude that Halperin discloses limitation 1[a].

12. Limitation 1[b]

438. Limitation 1[b] of the '250 patent reads:

at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

439. Halperin discloses this limitation as broken down in the paragraphs below.

At a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored.

440. In order to identify malware, Halperin discloses a centralized server that receives data about a virus from multiple computers. Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.***

(Ex. 1003 at [0047], emphasis added).

441. Notably, this disclosure from Halperin expresses that the same behavior must be detected at two or more computers in order for the server to determine that a virus may have infected the machines. Thus, the data received is data about the same or similar virus (or “object”) on two different computers.

442. Although the data received by the central server is behavioral, Halperin is clear that specific virus software (and not just general unwanted behavior) can be identified. One approach disclosed by Halperin is the use of “decoy” or “bait” elements on the system that, when accessed, trigger an alert about the accessing software. As Halperin explains, “In the method of FIG. 7 a number of virus detection and containment systems are implemented... each system may use ***different system decoys*** and/or monitor different correlation parameters... if one system ***detects a suspected virus using a specific decoy*** or correlation parameter, other systems may heighten their sensitivity level related to that decoy or correlation parameter.” (Ex. 1003 at [0061]; emphasis added). Figure 7 of Halperin, which illustrates part of this process, is reproduced hereinbelow:

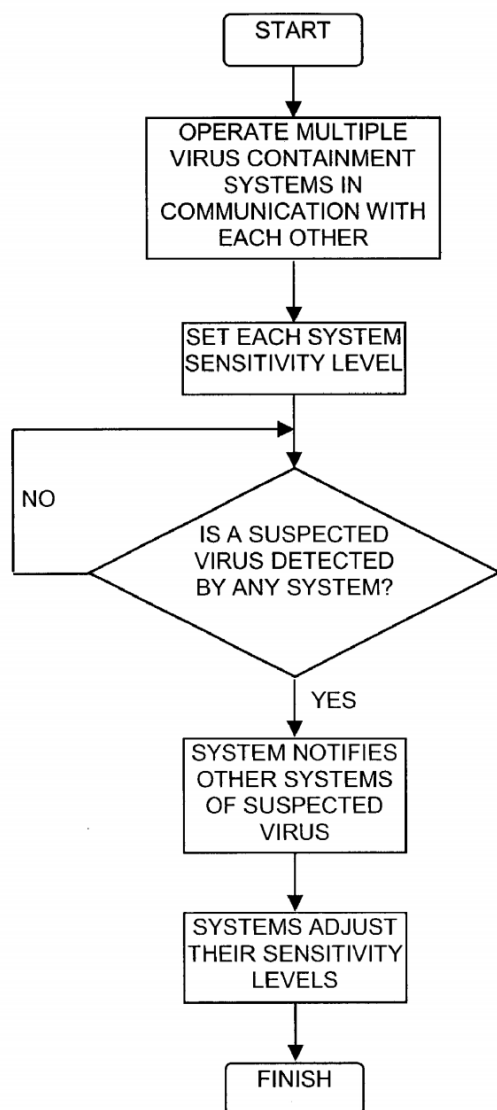


Fig. 7

443. In another example of detecting specific malware, Halperin discloses identifying communication messages that contain copies of the malicious software being transmitted over the network.

... the server may include a buffer or other mechanism whereby messages received from the computer are held, typically for a pre-defined delay period, prior to forwarding the messages to their intended recipients. In this way, should a computer virus send one or more *infected messages* to valid, non-decoy addresses before sending an infected message to a decoy address, *the infected messages to valid, non-decoy addresses that are still*

held at the server may be “quarantined” at the server and thus prevented, together with the infected message to a decoy address, from reaching their intended destinations.
(Ex. 1003 at [0058], emphasis added).

444. A POSITA would have understood “infected” messages to typically include the virus itself in some form or another.

the data including information about the behaviour of the object running on one or more remote computers;

445. The data transmitted to the server described above is explicitly about the behavior of the virus running on the other computers. In the same disclosure, Halperin describes the data received about the virus as “behavior.”

Additionally or alternatively, ***server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500*** using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, ***server 502 may then correlate the presence of target behavior detected at two or more of computers 500*** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers.
(Ex. 1003 at [0047], emphasis added).

446. Thus, I conclude that Halperin discloses limitation 1[b].

13. Limitation 1[c]

447. Limitation 1[c] of the '250 patent reads:

determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;

448. Halperin discloses determining at the centralized server if behavior data received from computers on the network indicates the presence of malicious software.

449. As I explained with reference to limitation 1[b], Halperin uses target behavior to detect computer viruses. Halperin states, “server 502 may then correlate the presence of target behavior detected at two or more of computers 500 in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior ***as an indication that a computer virus may have infected those computers.***” (Ex. 1003 at [0047], emphasis added). The detecting of malware is necessary step in order to mitigate or stop the malicious software from causing harm. In Halperin, the mitigation is described as virus containment: “Upon detecting a suspicious behavior pattern server 502 may initiate one or more virus containment actions...” (Ex. 1003 at [0057]).

450. Thus, I conclude that Halperin discloses limitation 1[c].

14. Limitation 1[d]

451. Limitation 1[d] of the '250 patent reads:

classifying the computer object as malware when the data indicates that the computer object is malware;

452. Halperin discloses identifying the detected data as malware at the central server. I refer to my analysis of limitation 1[b] and 1[c], which I incorporate by reference. In this analysis, Halperin classifies the software as malware as evidenced by the “virus containment” disclosed such as the quarantine process.

453. In more detail, when Halperin detects potential malicious activity, it engages “virus containment” on suspect programs or objects. In doing so, Halperin is identifying the object as malware. For example, Halperin lists the following examples of virus containment in which objects are classified as malware:

- a. Suspending any or all messages sent by computer 100, thereby preventing messages sent by computer 100 from being forwarded to recipients.
- b. Forwarding messages that are addressed to a decoy address to a third party for analysis, such as a company or other body that produces anti-virus software.
- c. Notifying a user at computer 100 of the suspicious message activity.
- d. Notifying a system administrator that a virus may have been detected.
- e. Stopping all messages from being forwarded by server 108 to their intended destinations. Taking away all privileges that computer 100 has to access network 102 and/or rights to access shared network files or directories.
- f. Changing the delay period of all messages received by server 108, thus putting the entire network on “virus alert.”;
- g. Sending a command to network devices connected to network 102, Such as Switches or routers, to block all attempts by computer 100 to access network 102. This may be done, for example, by using SNMP commands.

(Ex. 1003 at [0024]-[0030]).

454. All of these virus containment elements identify potentially malicious computer objects such as software that sends messages or access the network. A POSITA would have also understood that the messages being analyzed may contain the malware, which is why messages can be analyzed by anti-virus software.

455. Thus I conclude that Halperin discloses limitation 1[d].

15. Limitation 1[e]

456. Limitation 1[e] of the '250 patent reads:

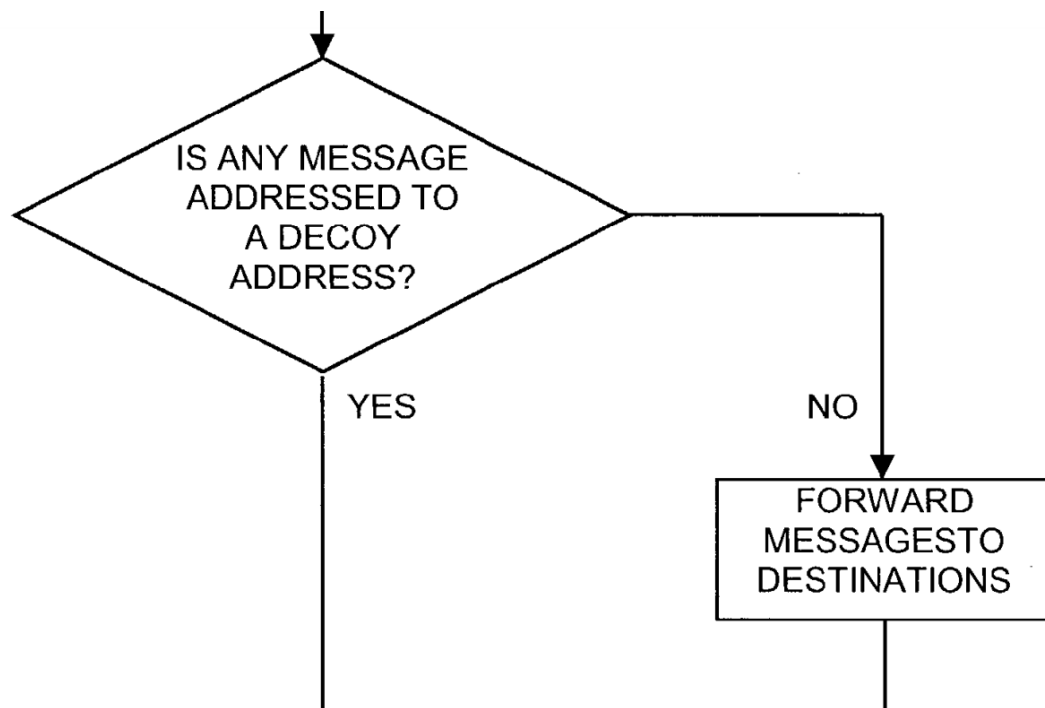
*when the determining does not indicate that the computer object is malware,
initially classifying the computer object as not malware;*

457. Halperin discloses having the central server not classifying “suspicious” software as malware in circumstances in which there is not enough evidence to do so. For example, Halperin discusses using a detection phase that analyzes behavior that includes a probability estimate. A customizable threshold determines whether or not to classify the behavior as malicious.

In the detecting phase the system logs every event of a sensor identifying a suspicious behavior on a specific computer into a central database. The logging preferably includes the type of event, the time it occurred and on which computer it occurred (if relevant). Either every predefined period of time or upon each event, the System queries the database for all events that took place during the last time slot. For example, if the time slot is 5 minutes, it gets all the events that took place in the past 5 minutes. These events are grouped by the sensor type and counted for that Sensor type. Thus, for example, it may be found that for the last 5 minute time slot 0 computers sent e-mails to a decoy address, 2 computers accessed a specific key in the Windows Registry and 5 computers accessed a decoy file. ***For each sensor, the probability of getting that result or a higher in a network without a virus is estimated based on the results in the learning database that were analyzed as described above.*** For example, if in the past 5 minutes there were 3 events of accessing the Windows Registry, the system checks what was the measured probability of having 3 or more computers with a Windows Registry event in the learning phase. ***It is then possible to estimate what is the combined probability that the different detectors would have the values that were measured assuming that there is no virus in the System.*** For example, all the results for the detectors for that time slot are converted to probabilities of having that value or higher as described above. The lowest 3 probabilities are taken, and a binomial distribution calculation is done for getting 3 successes out of K attempts when the probability of Success is equal to the highest of the 3 probabilities. ***The result is an estimate of the probability to get the result that was measured assuming that there is no virus in the system. Then the result is compared to a predefined threshold. If the calculated value is lower than the predefined threshold, the system can activate a virus alarm and possibly initiate virus containment actions. The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.*** The system additionally can have an option to have the operator enter information on whether at that time slot there is a centralized operation

that may look like a virus operation, e.g. a mass software installation in the network. In that case the sensors that may be activated by such an operation (e.g. Windows Registry sensors) may be disregarded for the time period defined by the operator for the centralized operation.
(Ex. 1003 at [0090], emphasis added)

458. An another example, Halperin has decision trees at the centralized server that categorize objects as either malicious or not malicious. For example, messages forwarded to a “decoy” address can be categorized as malicious and otherwise are categorized as not malicious. This process is visualized in Figure 2, reproduced hereinbelow.



459. A POSITA would have understood that messages forwarded to destinations are **not** categorized as malware. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[e]

16. Limitation 1[f]

460. Limitation 1[f] of the '250 patent reads:

automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data;

461. Halperin discloses having the central server identifying acceptable behavior for software running on the monitored computers. Inoue discloses creating a special sandbox for software based on acceptable behavior. The sandbox acts as the “mask” required by this limitation, as explained in more detail below.

462. Halperin discloses automatically monitoring system events and using “retrospective” information about known virus outbreaks to events that could be related to the virus and events that are not. A POSITA would have understood Halperin’s disclosure to refer to potentially unacceptable behavior (related to the virus) and acceptable behavior (when no viruses are present). Halperin explains:

In the learning phase ***all events are logged*** in a learning database similar to the database described above. This is done for a substantial period of time, for example, one week. In addition, the system gets retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system...

The learning database is sorted according to the time of each event logged. The events are grouped into non-overlapping time slots of a predefined length (e.g. 5 minutes). ***The learning database is then filtered*** according to the data logged in the virus outbreak database. ***Any time slot that is logged in the virus outbreak database as having an active virus in the system is taken out of the learning database.***

(Ex. 1003 at [0087]-[0088], emphasis added)

463. Inoue discloses a similar learning process for a specific piece of software. The software is run in a Java Virtual Machine (JVM) as part of a “training” session. During the training session, the calls to the operating system are detected as “profiling information.” The collected

profile data is encoded into a special sandbox that the program must run in thereafter. If the program attempts to deviate from the “trained” behavior, it is stopped.

In the first, **a sandbox profile is constructed** by running the program with an instrumented JVM. **During this training session, profiling information is recorded to the sandbox profile.** The sandbox is initially empty and grows during the training run by accumulating records for each unique behavior. Because nothing is added that isn’t observed, each sandbox is customized to a given program and context in which it is executed. **During sandbox execution, behavior that isn’t in the profile is considered anomalous.**

(Ex. 1005 at p. 3, emphasis added)

464. Thus, Inoue’s sandbox maps precisely with the concept of the mask as explained in the ’250 patent. As stated in the specification:

If an object is allowed to run on a remote computer 2... then operation of that object may be monitored within the parameters of the mask. The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on. Thus, the mask can give an overall picture of the expected “normal behaviour” of that object.

(Ex. 1001 at 14:58-15:4).

465. Moreover, Inoue describes that the mask is created automatically. I understand that in patent prosecution, Patent Owner distinguished the purported invention of the ’250 patent from prior art that used sandbox technology. Specifically, the Applicants differentiated the ’250 patent’s claims from the prior-art sandbox technology by alleging that the sandbox was “an isolated sandbox” “on a central computer” that was “**monitored by a human operator.**” (Ex. 1006, pp. 334, emphasis added). However, the sandboxing technology differentiated by Patent Owner is different. As I explained in the Technology Background, sandboxes **can** be used for observation and forensic analysis. This process, called detonation, can be automated but is also often done

with a “human-in-the-loop.” The idea is that bad software can be used within this sandbox and the effects studied. This is the type of sandbox used in Dozortsev, the prior art reference used by the examiner. (See, e.g., Ex. 1012 at [0040]).

466. Inoue, however, is not using this kind of sandbox. Inoue’s JVM (Java Virtual Machine) is not a “detonation” environment. JVMs (which existed before Inoue) are used to ensuring the safe operation of a Java program in an automatic and unsupervised fashion. Traditional JVMs, as described in Inoue and as is well known in the art, automatically enforce that no sensitive operation (e.g., system call) is permitted unless allowed by a pre-specified policy. The enforcement requires no human intervention. For example, Inoue explains, “The Security Manager sits on top of this infrastructure. All potentially dangerous actions such as I/O calls to the network or filesystem are guarded by a call to the Security Manager, which interprets the policy. A policy is a list permissions for each sandbox in the JVM. Interaction between different sandboxes is handled internally through stack introspection [34].” (Ex. 1005 at p. 2).

467. Inoue adds to this automated enforcement the **automated** development of a security policy. Security policies can, and often are, specified manually based on the analysis of a security operator. However, Inoue explains that the Java software in question can be run in a JVM during a training mode in which all operations are recorded (automatically) by the JVM itself. Repeating the previous citation to Inoue:

In the first, **a sandbox profile is constructed** by running the program with an instrumented JVM. **During this training session, profiling information is recorded to the sandbox profile.** The sandbox is initially empty and grows during the training run by accumulating records for each unique behavior. Because nothing is added that isn’t observed, each sandbox is customized to a given program and context in which it is executed. **During sandbox execution, behavior that isn’t in the profile is considered anomalous.**

(Ex. 1005 at p. 3, emphasis added)

468. The recording of the profile is automatic by the JVM and, without modification, is not even capable of permitting human observation. Inoue describes that the execution of the training environment requires a single modification to the normal execution of the program; that is, adding the flag “-profile <filename>”. (Ex. 1005 at p. 4). However, a POSITA would have understood that the execution of a file is automatable and there is no reason that the combination with Halperin would not launch this training session in an automated fashion. In fact, Inoue explicitly states that the entire purpose of the system was to create automated intrusion defense systems (IDS). “However, incorporating application-level information into an IDS is a nontrivial exercise, and *in the earlier projects*, the application was modified and the IDS *constructed by hand* [3, 36]. *Our approach automates the construction of an AppIDS* without modifying the application.” (Ex. 1005 at p. 1, emphasis added). It would have been obvious to a POSITA to generate Inoue’s sandbox on the central system after receiving information about the object from the remote computers disclosed by Halperin. Halperin relies on data from multiple systems that is transmitted to and received by the central computer in order for the central computer to determine if the software is good or bad. Moreover, because the software has appeared on multiple computers, it is most efficient to generate the sandbox once on the central computer and then transmit the sandbox out to all the computers that need to run the program.

469. A POSITA would also have recognized that the sandbox is generated in accordance with normal behavior determined from data transmitted by the remote computers. Although the training data of behavior for the object is generated by Inoue on the central computer, a POSITA would have known that this training data must match the behavior data transmitted by the remote computer, otherwise it would not run on the remote computers.

470. A POSITA would have been motivated to combine Halperin and Inoue. As explained in Section XI.A., a POSITA would be motivated to combine these two references in general. But specifically, it would have been obvious use Inoue as described in this section to build a sandbox with Halperin in order to extend Halperin's dynamic system. That is, Halperin discloses a system that learns as it goes, collecting data about "normal" and "abnormal" operations. Halperin, for example, discloses using increasing suspicious activity to eventually activate containment protocols.

In an alternative implementation of the buffer described above that is designed to reduce false alarms, should the server receive an invalid decoy message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified. During the increased delay period, ***should additional suspicious messages be received, or should other suspicious behavior be detected, if the user and/or system administrator who is authorized to do so has not indicated that the activity is not virus related, only then does the server perform one or more virus containment actions.*** If, however, during the increased delay period no other suspicious activity is detected, or if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.

(Ex. 1003 at [0059], emphasis added.)

471. Nevertheless, Halperin does identify that one of the problems of this approach is that there must be a trade off between false alarms (detecting a "good" program as a bad one) and not detecting malware at all. As Halperin explains, "... the [probability] result is compared to a predefined threshold. If the calculated value is lower than the predefined threshold, the system can activate a virus alarm and possibly initiate virus containment actions. ***The lower the threshold the less false alarms the system will have and the less sensitive it will be to identifying viruses.***" (Ex. 1003 at [0090], emphasis added).

472. Inoue offers another alternative to a simple false-alarm/less-detection trade-off. A program believed to be benign can have a “sandbox” wrapped around it limiting the software to its “good” behavior. That way, if the software “gets worse,” it is stopped dynamically. As Inoue explains, “We show our implementation in a Java Virtual Machine is both effective and efficient at stopping a backdoor and a virus, and ***has a low false positive rate.***” (Ex. 1005 at p. 1, emphasis added.)

473. A POSITA would have been motivated to add these protections to Halperin, preventing damage when malware is not detected and improving the false positive rates.

474. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[f].

17. Limitation 1[g]

475. Limitation 1[g] of the ’250 patent reads:

running said object on at least one of the remote computers;

476. Halperin discloses allowing software to run on remote computers until certain triggers are reached (e.g., a probability threshold exceeded). I discussed this disclosure in context of limitation 1[f] and I incorporate my analysis of this disclosure into this section by reference.

477. Inoue discloses a “sandbox” that allows software to run in a monitored container that enforces acceptable behavior based on training data. I discussed this disclosure in context of limitation 1[f] and I incorporate my analysis of this disclosure into this section by reference.

478. A POSITA would have been motivated to use Inoue’s sandbox to protect systems from software running on the remote computers that, although not yet determined to be viruses, had suspicious triggers. As I explained in reference to limitation 1[f], Halperin allows such software to run until it has a high enough confidence to decide to block it. The confidence required is configurable and the trade-off is false alarms for lack of detection. Using Inoue, it is possible

to allow the software to run with a profile of acceptable behavior. If any activities stray from this profile, the execution is stopped preventing damage to the remote systems. As Inoue explains, “We show our implementation in a Java Virtual Machine is both effective and efficient at stopping a backdoor and a virus, and ***has a low false positive rate.***” (Ex. 1005 at p. 1, emphasis added.)

479. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[g].

18. Limitation 1[h]

480. Limitation 1[h] of the '250 patent reads:

automatically monitoring operation of the object on the at least one of the remote computers;

481. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

482. Inoue discloses automatically monitoring software as it runs to ensure that its behavior does not stray outside of an authorized profile of approved operations. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference. I also explained why a POSITA would have been motivated to use Inoue’s disclosures in combination with Halperin’s teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

483. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[h].

19. Limitation 1[i]

484. Limitation 1[i] of the '250 patent reads:

allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;

485. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

486. Inoue discloses automatically monitoring software as it runs to ensure that its behavior does not stray outside of an authorized profile of approved operations. Inoue discloses a “sandbox” as a means to provide this monitoring and to shutdown the software if it misbehaves. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference. I also explained why a POSITA would have been motivated to use Inoue’s disclosures in combination with Halperin’s teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

487. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[i].

20. Limitation 1[j]

488. Limitation 1[j] of the ’250 patent reads:

disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and

489. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If

enough suspicious behavior is detected, the centralized server will classify the software as a virus. I incorporate my analysis of this disclosure into this section by reference.

490. Inoue discloses automatically monitoring software as it runs to ensure that its behavior does not stray outside of an authorized profile of approved operations. Inoue discloses a “sandbox” as a means to provide this monitoring and to shutdown the software if it misbehaves. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference. I also explained why a POSITA would have been motivated to use Inoue’s disclosures in combination with Halperin’s teachings. I also incorporate my analysis of the motivation to combine into this section by reference.

491. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[j].

21. Limitation 1[k]

492. Limitation 1[k] of the ’250 patent reads:

reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

493. Halperin discloses continuing to monitor automatically, at the centralized server, software running on the remote computers and collecting data about suspicious behaviors. As I explained with reference to limitation 1[f], Halperin collects suspicious behaviors over time. If enough suspicious behavior is detected, the centralized server will classify the software as a virus. In other words, Halperin initially classifies the software as **not malicious** but as more information is received (especially from multiple computers) it changes its classification to “virus.” I incorporate my analysis of this disclosure into this section by reference.

494. Inoue discloses automatically monitoring software as it runs to ensure that its behavior does not stray outside of an authorized profile of approved operations. Inoue discloses a

“sandbox” as a means to provide this monitoring and to shutdown the software if it misbehaves. I provided my analysis of this disclosure in the context of limitation 1[f] and I incorporate this analysis into this section by reference.

495. A POSITA would have been motivated to combine Inoue’s Sandbox with Halperin such that if the sandbox detected misbehavior it would alert Halperin’s centralized server. As I have already explained, Halperin’s central server collects data about behavior that is deemed suspicious. Halperin explains that it uses many “sensors” (each sensor typically running on one of the individual computers in the network) to collect the data: “In the embodiments described hereinabove for detecting suspicious activities in a computer network different detectors (e.g. file deletion and Windows Registry access) and decoys (e.g. e-mail decoy) were defined. These will all be referred to from this point onward as sensors.” (Ex. 1003 at [0086].)

496. It would have been obvious to a POSITA to use Inoue’s sandbox as a “detector” or “sensor.” A POSITA would have recognized that behavior that goes outside of the sandbox’s approved profile is suspicious and Halperin’s central server would be interested in the information. As Halperin explains, for “...the [malware] detecting phase the system logs **every event** of a sensor identifying a suspicious behavior on a specific computer into a central database.” (Ex. 1003 at [0090], emphasis added.)

497. As also explained previously, once Halperin has received additional suspicious information about software it will recategorize it as a virus and execute containment protocols. As previously described, Halperin discloses various methods for identifying suspicious activity and collecting data on behavior of software before it is identified as a virus or quarantined. (Ex. 1003 at [0061], [0090]). Halperin, as I have also explained, updates its operations as it obtains more information. For example, “[i]n the method of FIG. 7 a number of virus detection and containment

systems are implemented... different systems may have greater or lesser sensitivity [or threshold] levels, or simply different sensitivity levels by employing different Sensitivity parameters... each system may use different system decoys and/or monitor different correlation parameters... if one system ***detects a suspected virus using a specific decoy or correlation parameter, other systems may heighten their sensitivity level [e.g., threshold] related to that decoy or correlation parameter.***” (Ex. 1003 at [0061]; emphasis added).

498. As described regarding limitation 1[f], behavior may initially be classified as not malware despite being suspicious. Halperin discloses that if the suspicious behavior is correlated across ***multiple*** computers the classification can change (i.e., “reclassification”). Specifically, Halperin states: “[a]ctivity that cannot be confidently attributed to a virus when ***observed on one computer*** can be clearly identified as such ***when observed on several computers in a network.***” (Ex. 1003 at [0037]).

499. In another example, Halperin describes how identification of malware (viruses) can be group based. Once detection occurs in one group, other groups will be updated (i.e., the software is reclassified):

The present invention may be employed to identify suspicious activity occurring in multiple groups simultaneously. For example, if suspicious behavior is detected at a computing device, and similar suspicious behavior is also detected in various groups to which the computing device belongs, virus containment actions may be taken in each of the groups.
(Ex. 1003 at [0079]).

500. Thus, I conclude that Halperin in view of Inoue discloses limitation 1[k].

C. Claim 2 is rendered obvious by Halperin in view of Inoue

501. Claim 2 of the ’250 patent, which depends from Claim 1, reads:

A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of: executable

instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

502. I have already explained that Halperin in view of Inoue render obvious Claim 1. Halperin discloses the additional requirements of Claim 2.

503. Halperin discloses that the target behavior, which is data about the potential virus sent from multiple computers to a central server, includes:

- a. Sending messages to more than a predefined number of users during a predefined time.
- b. Sending messages not as a result of a direct user interaction with the Graphic User Interface (GUI) of the message software, but rather as the result of a directive from a software application.
- c. Modifying operating system files such as the Microsoft Windows registry.
- d. Deleting more than a predefined number of files on the computer's hard disk during a predefined period of time.
- e. Loading a new software application into the computer's RAM.
- f. Sending a file attached to a message several times from the same user.
- g. Sending a file attachment of a specific type (e.g., .exe, .doc, zip).
- h. Attempting to contact previously unused or unknown IP addresses or IP sockets.

(Ex. 1003 at [0038]-[0046]).

504. All of these behaviors are examples of events initiated by the object when the object is running on the respective remote computers.

505. Thus, I conclude that Halperin discloses the additional requirements of Claim 2.

D. Claim 3 is rendered obvious by Halperin in view of Inoue

506. Claim 3 of the '250 patent, which depends from Claim 1, reads:

A method according to claim 1, wherein determining identifies relationships between the object and other objects.

507. I have already explained that Halperin in view of Inoue render obvious Claim 1. Halperin discloses the additional requirements of Claim 3.

508. Halperin discloses having the centralized server determine if software is potentially a virus by correlating together data about the behavior of software across multiple machines and its interaction with other software and systems. Halperin states:

After collecting information regarding target behavior detected at two or more of computers 500, server 502 may then correlate the presence of target behavior detected at two or more of computers 500 in order to determine whether the correlated target behavior corresponds to a predefined Suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers. Any known behavior correlation techniques may be used, Such as identifying the same activity in different computers at about the same time, or by identifying repeating patterns of data within the memories of two or more computers.
(Ex. 1003 at [0047]).

509. Halperin provides multiple examples of identifying the relationships between the detected objects:

- a. A certain percentage of the computers in the network sending more than 10 messages per minute in the last 5 minutes,

- b. A certain percentage of the computers in the network sending messages not initiated via the message GUI in the last 1 minute;
- c. A certain percentage of the computers in the network deleting more than 10 files in the last 1 minute;
- d. A certain percentage of computers in the network deleting a file by the same name within the last 1 hour.
- e. A certain percentage of the computers in the network deleting a file with the same name in the last 1 minute;
- f. A certain percentage of the computers in the network to which changes to the Microsoft Windows Registry occurred in the last 1 minute;
- g. A certain percentage of the computers in the network Sending the same file attachment via a message in the last 15 minutes,
- h. A certain percentage of the computers in the network sending file attachments via one or more messages in the last hour where each of the files includes the same string of bits,
- i. A certain percentage of the computers in the network having an unusual level of correlation of data between files sent as attachments. For example, since viruses known as "polymorphic viruses" may change their name as they move from one computer to another, one way to identify Such viruses is to identify attachments that have the same or similar data, whether or not they have the same name.

(Ex. 1003 at [0048]-[0056]).

510. Each of these examples identifies at least a relationship between the object on one computer and the object on another because “certain percentage” refers to these objects behaving in the same way. Sending a message, deleting files, and modifying the registry also establish relationships between the monitored object and other objects where the other objects are messages, files, and registry entries. Attachments, which are special categories of files included in messages, are also other objects being related to the monitored object.

511. Thus, I conclude that Halperin discloses the additional requirements of Claim 3.

E. Claim 4 is rendered obvious by Halperin in view of Inoue

512. Claim 4 of the '250 patent, which depends from Claim 3, reads:

A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

513. I have already explained that Halperin in view of Inoue render obvious Claim 3. Halperin discloses the additional requirements of Claim 4.

514. Halperin discloses sharing information between computers, and aggregated at the central server, to identify malware spread from one computer to another. For example, Halperin discloses using attachments that are identified as viruses to identify other attachments as viruses. “For example, since viruses known as ‘polymorphic viruses’ may change their name as they move from one computer to another, one way to identify such viruses is to identify attachments that have the same or similar data, whether or not they have the same name.” (Ex. 1003 at [0056]). These attachment objects are related to each other by having the same or similar data. Identifying one as malware is used to identify the others as malware.

515. In another example, Halperin discloses that identifying a characteristic of malware in one set of computers can be used to identify that malware in another group of computers. According to Halperin, “if a change to a certain software variable is used to identify the malicious

software in one group, the same change may be monitored for in neighboring groups.” (Ex. 1003 at [0069]).

516. In a third example, Halperin also explains that a “checksum” (which is a code that identifies data) is generated for malware on one machine, that same code can be forwarded to other machines for use in identifying the same malware. “A characteristic code may be generated for the attachment that distinguishes it from other such attachments. This may be done using well known ‘checksum’ algorithms. The checksum may then be sent to neighboring computers within the group and to computers within neighboring groups which may then use the checksum to identify suspicious malicious software upon arrival at these computers.” (Ex. 1003 at [0073]).

517. In all of these examples, Halperin discloses the use of one identified virus/malware to identify the same or similar viruses on other machines.

518. Thus, I conclude that Halperin discloses the additional requirements of Claim 4.

F. Claim 5 is rendered obvious by Halperin in view of Inoue

519. Claim 5 of the ’250 patent, which depends from Claim 3, reads:

A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

520. I have already explained that Halperin in view of Inoue render obvious Claim 3. Halperin discloses the additional requirements of Claim 5.

521. As I explained with reference to Claim 3, Halperin discloses identifying the malware/virus objects on the remote computers. A POSITA would have understood that this means “stored” on the remote computers. I incorporate my analysis of Claim 3 into this section by reference.

522. Thus, I conclude that Halperin discloses the additional requirements of Claim 5.

G. Claim 12 is rendered obvious by Halperin in view of Inoue

523. Claim 12 of the '250 patent, which depends from Claim 1, reads:

A method according to claim 1, comprising if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future.

524. I have already explained that Halperin in view of Inoue render obvious Claim 1. Halperin in view of Inoue discloses the additional requirements of Claim 12.

525. Halperin discloses that a fundamental part of the system is to monitor at the centralized server suspicious behavior of remote computers and use detected suspicious behavior to identify viruses. But Halperin also recognizes that, in order to reduce false alarms, there must be a means to “clear” the behavior of being suspicious. One means of doing this is simply to have an administrator indicate that the activity is safe. Halperin explains that “...to reduce false alarms, should the server receive an invalid decoy message, or should suspicious behavior be detected for multiple computers, the buffer delay period may be increased by a predetermined amount of time, and users may be notified... if the user and/or system administrator who is authorized to do so has indicated that the activity is not virus related, the delay period may be reduced to its previous level and no virus containment action is performed.” (Ex. 1003 at [0059]).

526. Halperin also teaches that quarantines provide an opportunity for a “suspicious” object to be cleared. “The server may also notify a system administrator of the quarantined messages who may then check the quarantined to determine whether or not the messages were

indeed sent by a computer virus and either allow them to be forwarded to their intended recipients as is...” (Ex. 1003 at [0058]).

527. As I noted above, in addition to identifying suspicious behavior, Halperin also discloses identifying normal behavior. Learning “normal” behavior is necessary for identifying “abnormal” behavior. Halperin teaches that “[o]nce the groups are defined, it is possible to define and measure different parameters that are indicative of the methods of operation within and between the groups. Over time the characteristic values of these parameters during normal operation may be learned.” (Ex. 1003 at [0071]). As I have already explained, Halperin is an automated system that updates itself as it obtains more information. As I noted with respect to limitation 1[k], Halperin explains that “[a]ctivity that cannot be confidently attributed to a virus when **observed on one computer** can be clearly identified as such **when observed on several computers in a network**.” (Ex. 1003 at [0037], emphasis added). Thus, Halperin adjusts automatically as more information becomes available.

528. I also explained this with respect to limitation 1[b]. Specifically, Halperin discloses receiving data about a virus from **multiple computers**. The classification is adjusted as the information becomes available from the multiple sources. Halperin explains:

Additionally or alternatively, server 502 may be configured with such target behavior profiles and may detect associated target behavior at computers 500 using conventional techniques. After collecting information regarding target behavior detected at two or more of computers 500, **server 502 may then correlate the presence of target behavior detected at two or more of computers 500** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior **as an indication that a computer virus may have infected those computers**. (Ex. 1003 at [0047], emphasis added).

529. It would have been obvious to a POSITA to also use the behavior of objects to **continuously** identify normal patterns of behavior and not just once. A POSITA would have found

it obvious to use this information in order to retrain the system, as directed by the administrator, as new information becomes available.

530. Inoue, as explained with reference to Claim 1, also discloses a learning process for a specific piece of software. The software is run in a Java Virtual Machine (JVM) as part of a “training” session. During the training session, the calls to the operating system are detected as “profiling information.” The collected profile data is encoded into a special sandbox that the program must run in thereafter. If the program attempts to deviate from the “trained” behavior, it is stopped.

In the first, ***a sandbox profile is constructed*** by running the program with an instrumented JVM. ***During this training session, profiling information is recorded to the sandbox profile.*** The sandbox is initially empty and grows during the training run by accumulating records for each unique behavior. Because nothing is added that isn’t observed, each sandbox is customized to a given program and context in which it is executed. ***During sandbox execution, behavior that isn’t in the profile is considered anomalous.***

(Ex. 1005 at p. 3, emphasis added).

531. It would have been obvious to a POSITA to update the training of the sandbox as part of the continuous retraining discussed above. A POSITA would have found it obvious to “quarantine” programs that violated their sandbox or otherwise notify an administrator of the violation, as disclosed by Halperin. It would have been equally obvious that the administrator would use continuous training to update rules if the software was not, in fact, malicious.

532. As I explained with respect to limitation 1[k], Inoue and Halperin emphasize an automated process. It would have been obvious to a POSITA to have the continuous updates also be automatic.

533. It would have been obvious to combine Halperin and Inoue as discussed in Section XI.A. A POSITA would have recognized that the use of continuous training of Inoue in

combination with Halperin's system would further reduce false positives and further enable the administrator to respond to identified false positives.

H. Claim 13 is rendered obvious by Halperin in view of Inoue

534. Halperin in view of Inoue discloses all of the limitations of Claim 13.

1. Limitation 13[a]

535. Limitation 13[a] of the '250 patent reads:

Apparatus for classifying a computer object as malware, the apparatus comprising

536. See analysis of limitation 1[a]. As I explained with reference to Claim 1, Halperin discloses a centralized server that functions as a "base computer" performing the operations recited in Claim 13 and its dependent claims. The centralized server is thus "constructed and configured" to perform the operations of these claims.

537. Halperin's discloses an apparatus. Specifically, Halperin's "computer virus detection system" is an apparatus as shown in the Figure 1 and Figure 5 reproductions below (see also Ex. 1003 at [0036]):

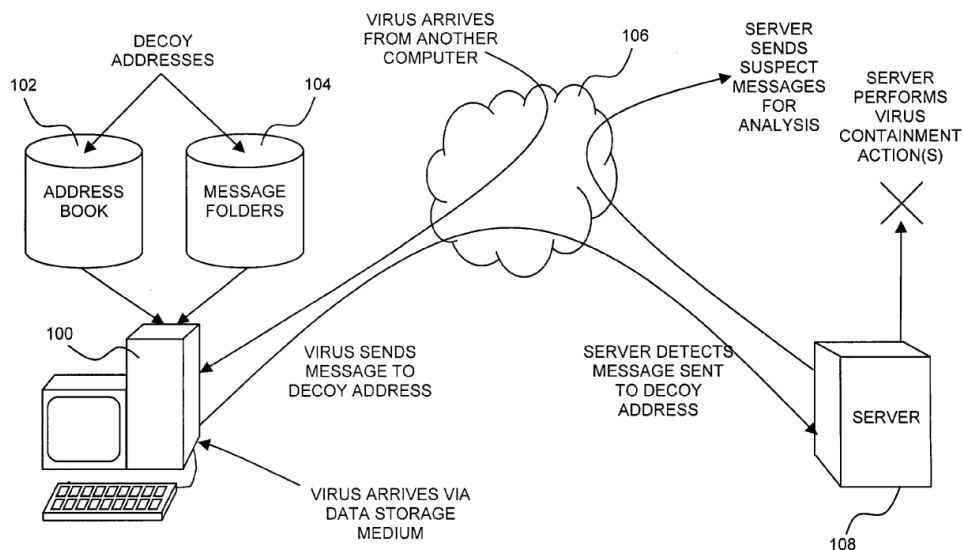


Fig. 1

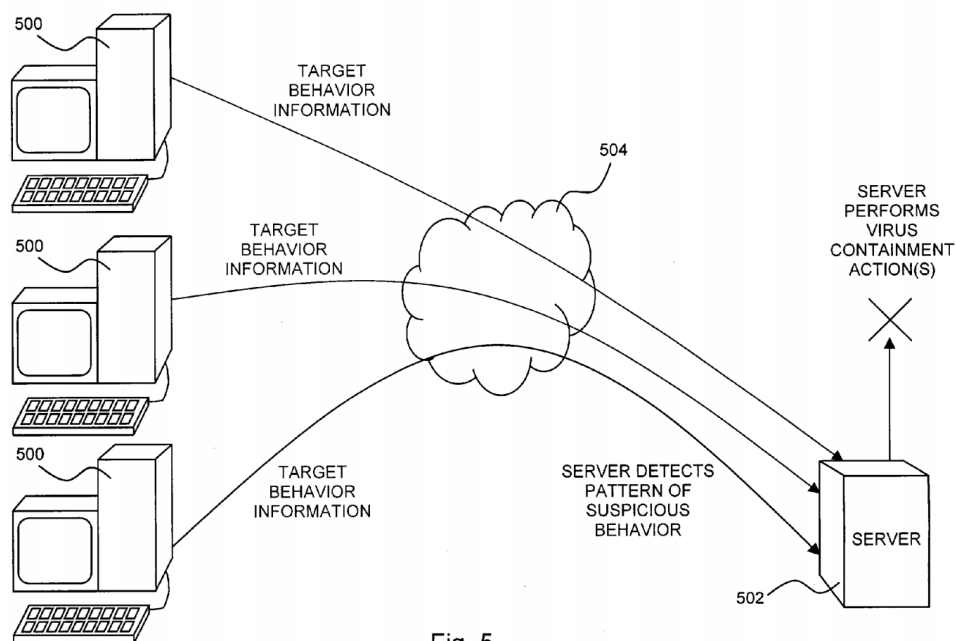


Fig. 5

538. Moreover, Halperin discloses, “While the methods and apparatus disclosed herein may or may not have been described with reference to specific hardware or software, it is appreciated that the methods and apparatus described herein may be readily implemented in hardware or Software using conventional techniques.” (Ex. 1003 at [0092]).

539. Thus, Halperin discloses limitation 13[b].

7. Limitation 13[b]

540. Limitation 13[b] of the '250 patent reads:

a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

541. See analysis of limitation 1[b].

542. Thus, I conclude that Halperin discloses limitation 13[b].

8. Limitation 13[c]

543. Limitation 13[c] of the '250 patent reads:

the base computer being constructed and arranged to determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,

544. See analysis of limitation 1[c].

545. Thus, I conclude the Halperin discloses limitation 13[c].

9. Limitation 13[d]

546. Limitation 13[d] of the '250 patent reads:

the base computer being constructed and arranged to classify the computer object as malware when the base computer determines that the data indicates that the computer object is malware,

547. See analysis of limitation 1[d].

548. Thus, I conclude that Halperin discloses limitation 13[d].

10. Limitation 13[e]

549. Limitation 13[e] of the '250 patent reads:

wherein the base computer is constructed and arranged to automatically generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data, wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers, the object being allowed to continue to run when behaviour of the object is permitted by the mask, the object not being permitted to run when the actual

monitored behaviour of the object extends beyond that permitted by the mask, and the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.

550. See analysis of limitations 1[e]-1[f].

551. Thus, I conclude that Halperin in view of Inoue discloses limitation 13[e].

11. Limitation 13[f]

552. Limitation 13[f] of the '250 patent reads:

wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers, the object being allowed to continue to run when behaviour of the object is permitted by the mask, the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.

553. See analysis of limitations 1[g]-1[k].

554. Thus, I conclude that Halperin in view of Inoue discloses limitation 13[f].

555.

I. Claim 14 is rendered obvious by Halperin in view of Inoue

556. Claim 14 of the '250 patent, which depends from Claim 13, reads:

Apparatus according to claim 13, the data includes one or more of executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object;

the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

557. See analysis of Claim 2.

558. Thus, I conclude that Halperin in view of Inoue discloses Claim 2.

J.Claim 15 is rendered obvious by Halperin in view of Inoue

559. Claim 15 of the '250 patent, which depends from Claim 13 reads:

Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the determining identifies relationships between the object and other objects.

560. See analysis of Claim 3.

561. Thus, I conclude that Halperin in view of Inoue discloses Claim 15.

K. Claim 16 is rendered obvious by Halperin in view of Inoue

562. Claim 16 of the '250 patent, which depends from Claim 15 reads:

Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

563. See analysis of Claim 4.

564. Thus, I conclude that Halperin in view of Inoue discloses Claim 16.

L. Claim 17 is rendered obvious by Halperin in view of Inoue

565. Claim 17 of the '250 patent, which depends from Claim 15, reads:

Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least Sonic of said remote computers.

566. See analysis of Claim 5.

567. Thus, I conclude that Halperin in view of Inoue discloses Claim 17.

M. Claim 24 is rendered obvious by Halperin in view of Inoue

568. Claim 24 of the '250 patent, which depends from Claim 13, reads:

A method according to claim 13, wherein, if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, the base computer is arranged to compare the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects the base computer is arranged to allow said behaviour for the object and to modify the mask to allow that behaviour for that object in future

569. See analysis of Claim 12.

570. Thus, I conclude that Halperin in view of Inoue discloses Claim 24.

XIII. THERE ARE NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

571. I am not aware of any objective indicia of non-obviousness related specifically to the '250 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 '250 patent.

XIV. CONCLUSION

572. 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 March 2, 2023

By: 
Seth James Nielson