

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT INC.
Patent Owner

Case No. IPR2023-00124
Patent No. 9,578,045

DECLARATION OF DR. WENKE LEE

TABLE OF CONTENTS

I.	INTRODUCTION	6
A.	MATERIALS REVIEWED	6
B.	BACKGROUND AND QUALIFICATIONS	8
II.	LEGAL FRAMEWORK	11
III.	OPINION.....	18
A.	OVERVIEW OF THE '045 PATENT	18
B.	OVERVIEW OF THE '045 PATENT'S FILE HISTORY	21
C.	LEVEL OF A PERSON HAVING ORDINARY SKILL IN THE ART	24
D.	BACKGROUND OF THE TECHNOLOGY.....	26
E.	CLAIM CONSTRUCTION	39
F.	SUMMARY OF THE PRIOR ART REFERENCES	39
1.	<i>Morris</i>	39
2.	<i>Van Oorschot</i>	46
3.	<i>Capalik</i>	50
G.	OPINIONS REGARDING GROUND 1: CLAIMS 1-2, 4, 8-10, 12, AND 16 ARE RENDERED OBVIOUS BY MORRIS OR ALTERNATIVELY, CLAIMS 1-4, 8-10, 12, AND 16 ARE RENDERED OBVIOUS BY MORRIS IN COMBINATION WITH VAN OORSCHOT	56
1.	<i>Claim 1</i>	57
2.	<i>Claim 2 – The method of claim 1, wherein the information associated with the one or more of the first object and the originating object is obtained from a community database; and wherein the event line is transmitted to an end recipient.</i>	83
3.	<i>Claim 3 – The method of claim 1, further comprising: obtaining an event priority based on one or more of the generated context and the global perspective for the at least one event, wherein the event line further includes the event priority for the at least one event.....</i>	86
4.	<i>Claim 4 – The method of claim 1, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.</i>	97
5.	<i>Claim 8</i>	97

6.	<i>Claim 9 – The system of claim 8, wherein the at least one of the one or more events is executed by the processor.....</i>	<i>101</i>
7.	<i>Claim 10 – The system of claim 8, further comprising: a global perspective information server; and an end recipient device, wherein the information associated with the one or more of the first object and the originating object is obtained from the global perspective information server and the sensor agent transmits the event line to the end recipient device.....</i>	<i>101</i>
8.	<i>Claim 12 – The system of claim 8, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.....</i>	<i>104</i>
9.	<i>Claim 16.....</i>	<i>104</i>
H.	OPINIONS REGARDING GROUND 2: CLAIMS 1-2, 4, 7-10, 12, AND 15-18 ARE RENDERED OBVIOUS BY CAPALIK OR ALTERNATIVELY, CLAIMS 1-4, 7-10, 12, AND 15-18 ARE RENDERED OBVIOUS BY CAPALIK IN COMBINATION WITH VAN OORSCHOT	107
1.	<i>Claim 1.....</i>	<i>107</i>
2.	<i>Claim 2 – The method of claim 1, wherein the information associated with the one or more of the first object and the originating object is obtained from a community database; and wherein the event line is transmitted to an end recipient.</i>	<i>131</i>
3.	<i>Claim 3 – The method of claim 1, further comprising: obtaining an event priority based on one or more of the generated context and the global perspective for the at least one event, wherein the event line further includes the event priority for the at least one event.....</i>	<i>133</i>
4.	<i>Claim 4 – The method of claim 1, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.....</i>	<i>136</i>
5.	<i>Claim 7 – The method of claim 4, further comprising: heuristically displaying information identifying the first object performing the action, information identifying the</i>	

	<i>action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.....</i>	<i>137</i>
6.	<i>Claim 8.....</i>	<i>140</i>
7.	<i>Claim 9 – The system of claim 8, wherein the at least one of the one or more events is executed by the processor.....</i>	<i>147</i>
8.	<i>Claim 10 – The system of claim 8, further comprising: a global perspective information server; and an end recipient device, wherein the information associated with the one or more of the first object and the originating object is obtained from the global perspective information server and the sensor agent transmits the event line to the end recipient device.....</i>	<i>147</i>
9.	<i>Claim 12 – The system of claim 8, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.....</i>	<i>148</i>
10.	<i>Claim 15 – The system of claim 12, wherein the end recipient heuristically displays information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.....</i>	<i>148</i>
11.	<i>Claim 16.....</i>	<i>151</i>
12.	<i>Claim 17 – The system of claim 16, further comprising a global perspective information server including: a processor; a communication interface; data storage; and a global perspective module, wherein the global</i>	

perspective module is operable to: receive the event line from the device; obtain a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information; assemble a second event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmit the assembled second event line utilizing the communication interface to an end recipient..... 153

13. *Claim 18 – The system of claim 17, wherein the end recipient heuristically displays information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object. 155*

IV. CONCLUSION..... 155

I, Wenke Lee, declare the following:

I. INTRODUCTION

1. I have been retained by counsel for Petitioner as a technical expert in the above-captioned case. Specifically, I have been asked to render certain opinions in regard to the IPR petition with respect to U.S. Patent No. 9,578,045 (the “’045 Patent”) (Ex. 1001). I understand that the Challenged Claims are claims 1-4, 7-10, 12, and 15-18 and my opinions are limited to those claims. My compensation in this matter is not based on the substance of my opinions or the outcome of this matter. I have no financial interest in Petitioner.

A. Materials Reviewed

2. In reaching my opinions in this matter, I have reviewed the following materials:

- Exhibit 1001 – U.S. Patent No. 9,578,045 to Jaroch et al.
- Exhibit 1002 – File History of U.S. Patent No. 9,578,045
- Exhibit 1004 – U.S. Publication No. 2013/0298244 to Kumar et al.
- Exhibit 1005 – U.S. Publication No. 2007/0016953 to Morris et al.
- Exhibit 1006 – U.S. Patent No. 8,087,087 to Van Oorschot et al.
- Exhibit 1007 – U.S. Patent Pub. No. 2010/0077481 to Polyakov et al.
- Exhibit 1008 – U.S. Publication No. 2011/0321166 to Capalik et al.
- Exhibit 1009 – U.S. Publication No. 2012/0260340 to Morris et al.
- Exhibit 1010 – U.S. Patent No. 10,257,224 to Jaroch et al.

- Exhibit 1012 – Natarajan Meghanathan, Sumanth Reddy Allam, and Loretta A. Moore, *Tools and Techniques for Network Forensics*, International Journal of Network Security & Its Applications (IJNSA), Vol. 1., No. 1 (April 2009)
- Exhibit 1013 – SANS Institute, *Host- vs. Network-Based Intrusion Detection Systems* (2000-2005)
- Exhibit 1014 – Securosis LLC, *Evolving Endpoint Malware Detection: Dealing with Advanced and Targeted Attacks*, Version 1.3 (July 12, 2012)
- Exhibit 1015 – Yanfang Ye, Weiwei Zhuang, Tao Li, Egemen Tas, Umesh Gupta, Shenghuo Zhu, Melih Abdulhayogiu, *Combining File Content and File Relations for Cloud Based Malware Detection* (August 2011)
- Exhibit 1016 – U.S. Patent No. 6,944,772 to Dozortsev
- Exhibit 1017 – U.S. Patent No. 6,772,363 to Chess et al.
- Exhibit 1018 – WO 2002/033525 to Shyne-Song Chuang
- Exhibit 1019 – EP 1,549,012 to Kristof De Spiegeleer
- Exhibit 1020 – EP 1,280,040 to Alexander James Hinchliffe et al.
- Exhibit 1021 – Oracle, Java Documentation, *The Java Tutorials, What is a Path? (And Other File System Facts)* (available at the following link <https://docs.oracle.com/javase/tutorial/essential/io/path.html>) (last accessed October 2022)
- Exhibit 1022 – U.S. Patent Publication No. 2004/0143753 to Hernacki et al.
- Exhibit 1023 – U.S. Patent Publication No. 2011/0083180 to Mashevsky et al.
- Exhibit 1024 – U.S. Patent No. 8,429,746 to Alen Capalik
- Exhibit 1025 – *Server Hardware, Software and Architecture* by Abeer El Hadi El Zein El Nahas, University of Khartoum, Electrical and Electronics Engineering Dept. (Feb. 2008)

- Ex. 1026 – Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee, *BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation*, In Proceedings of The 16th USENIX Security Symposium (August 2007)
- Ex. 1027 – Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee, *Understanding the Prevalence and Use of Alternative Plans in Malware with Network Games*, In Proceedings of The 27th Annual Computer Security Applications Conference (December 2011)
- Ex. 1028 – Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster, *Building a Dynamic Reputation System for DNS*, In Proceedings of The 19th USENIX Security Symposium, (August 2010)
- Ex. 1029 – Long Lu, Vinod Yegneswaran, Phil Porras, and Wenke Lee, *BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections*, In Proceedings of The 17th ACM Conference on Computer and Communications Security (CCS), Chicago, IL (October 2010)
- Ex. 1030 – Plaintiffs’ Markman Opposition Brief (Dkt. 98)
- Ex. 1031 – Defendants’ Markman Opening Brief (Dkt. 86)

B. Background and Qualifications

3. I have summarized in this section my educational background, career history, and other qualifications relevant to this matter. I have also included a current version of my curriculum vitae, which is attached as Appendix A.

4. I received my bachelor’s degree in Computer Science from Sun Yat-Sen University in China in 1988. I received my master’s degree in Computer Science from The City College of New York in 1990, and my doctorate in Computer Science from Columbia University in New York in 1999.

5. I am now a Professor and the John P. Imlay Jr. Chair in the School of Cybersecurity and Privacy in the College of Computing at the Georgia Institute of Technology (Georgia Tech). I am knowledgeable and familiar with computer science and cybersecurity and privacy in general, and more specifically, malware analysis and detection, security of mobile ad-hoc networks, VPNs and network intrusion detection, adversarial machine learning, and adaptive analysis framework for advanced persistent threats.

6. As shown in my CV, I have authored several articles and books on computer science related to computer system monitoring, program analysis and software security, network security, intrusion and anomaly detection systems, VPNs, and mobile ad-hoc network security.

7. I have published extensively with about 200 refereed journal articles, refereed conference proceedings, refereed workshop proceedings, and books. My research has been recognized by many awards for contributions in wired and mobile network security, intrusion and botnet detection, malware analysis, network and software security, and machine-learning approaches for security analytics. In 2002, I was awarded a National Science Foundation Career Award. In 2015, I was awarded the Internet Defense Prize by Facebook and USENIX for work on stopping emerging attack vectors. I have also received several Best Paper Awards and a Test-of-Time Award in the past several years. I have been recognized as a pioneer in applying

machine-learning based to intrusion detection, and in 2017, I was selected as a fellow of the Association for Computing Machinery (ACM), in 2019, I received an Outstanding Innovation Award from the ACM, and in 2021, I was elected as a fellow of the Institute of Electrical and Electronics Engineers (IEEE).

8. I have received substantial funding from the US government agencies, industry, and foundations, and as a result, I and my students have developed innovative approaches for system monitoring, software security, malware analysis, network security, mobile network security, intrusion detection, and machine learning. In particular, I have won many research grants in the field of program analysis and systems and software security. For instance, from 6/26/2015-6/26/2019, I was the principal investigator (PI) for a \$4.19M Defense Advanced Research Project Agency (DARPA) project on tracking and analyzing software and computer events to detect cyberattacks. From 8/1/2016-7/31/2019, I was a co-PI for an Army Research Office project on proactive strategies for cyber defense. From 6/1/2016-5/31/2021, I was a co-PI on an Office of Naval Research project developing an analytical framework for actionable defense against Advanced Persistent Threats.

9. My research has resulted in many oft-cited academic papers in wired and mobile network security, intrusion detection, botnet detection, and malware analysis, as well as technologies adopted by the research community and industry.

For example, I co-founded Damballa Inc. to commercialize botnet detection products based on technologies originally developed by me and my students.

10. In sum, I have extensive experience both as a developer and a researcher relating to system monitoring, software security, attack detection, and network security.

II. LEGAL FRAMEWORK

11. I am a technical expert and do not offer any legal opinions. However, counsel has informed me as to certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

12. I have been informed that claim terms are to be given the meaning they would have to a person having ordinary skill in the art at the time of the invention, taking into consideration the patent, its file history, and, secondarily, any applicable extrinsic evidence (e.g., dictionary definitions).

13. I have also been informed that the implicit or inherent disclosures of a prior art reference may anticipate the claimed invention. Specifically, if a person having ordinary skill in the art at the time of the invention would have known that the claimed subject matter is necessarily present in a prior art reference, then the prior art reference may “anticipate” the claim. Therefore, a claim is “anticipated”

by the prior art if each and every limitation of the claim is found, either expressly or inherently, in a single item of prior art.

14. I have also been informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art. I have been informed that a conclusion of obviousness may be founded upon more than a single item of prior art. I have been further informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non-obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person having ordinary skill in the relevant art at the time the patent was filed.

15. In considering whether certain prior art renders a particular patent claim obvious, counsel has informed me that I can consider the scope and content of the prior art, including the fact that one of skill in the art would regularly look to the disclosures in patents, trade publications, journal articles, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials

from industry conferences, as examples. I have been informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be “analogous art” to the claimed invention. I have been informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). In order for a reference to be “reasonably pertinent” to the problem, it must logically have commended itself to an inventor's attention in considering his problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references that my opinions in this IPR are based upon are well within the range of references a person having ordinary skill in the art would consult to address the type of problems described in the Challenged Claims.

16. I have been informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that a combination of multiple items of prior art renders a patent claim obvious when there was an apparent reason for one of ordinary skill in the art, at the time of the invention, to combine the prior art, which can include,

but is not limited to, any of the following rationales: (A) combining prior art methods according to known methods to yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

17. I am informed that the existence of an explicit teaching, suggestion, or motivation to combine known elements of the prior art is a sufficient, but not a necessary, condition to a finding of obviousness. This so-called “teaching suggestion-motivation” test is not the exclusive test and is not to be applied rigidly in an obviousness analysis. In determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. Instead, the important consideration is the objective reach of the claim. In other words, if the claim extends to what is obvious, then the claim is

invalid. I am further informed that the obviousness analysis often necessitates consideration of the interrelated teachings of multiple patents, the effects of demands known to the technological community or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the art. All of these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent.

18. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the challenged claim need not be sought out because it is appropriate to take account of the inferences and creative steps that a person of ordinary skill in the art would employ. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem. I am informed that common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond their primary purposes.

19. I also am informed that the fact that a particular combination of prior art elements was “obvious to try” may indicate that the combination was obvious

even if no one attempted the combination. If the combination was obvious to try (regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an invention. I am informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious.

20. I am informed that for a patent claim to be obvious, the claim must be obvious to a person of ordinary skill in the art at the time of the invention. I am informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

21. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may be said to teach away when a person of ordinary skill in the relevant art, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent

applicant. In general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. I also am informed, however, that a reference does not teach away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed.

22. I am informed that the final determination of obviousness must also consider "secondary considerations" if presented. In most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in view of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; (h) teaching away from the invention in the prior art.

23. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. While I understand that the Patent Owner here has not offered any secondary considerations at this time, I will supplement my opinions if the Patent Owner raises secondary considerations during this proceeding.

III. OPINION

A. Overview of the '045 Patent

24. The '045 Patent relates to “[m]ethods and apparatuses for providing forensic visibility” and more specifically, “methods and systems for providing forensic visibility into client systems having full audit trails[.]” *'045 Patent*, 1:15-18. Per the '045 Patent, “[n]etwork forensics generally relates to the capture, recording, and analysis of network events in order to discover the source of security attacks or other problem incidents.” *Id.*, 1:22-24. The '045 Patent explains that “existing network forensic tools, such as networks sniffers and packet capturing tools” have been used to forensically analyze network events, but notes these “tools may be able to detect and capture information associated with communication sessions, such as hostnames, ports, protocols, IP addresses, etc.; however, such tools tend to receive network events outside of a host or system.” *Id.*, 1:34-46 (further

noting these “network edge devices have no ability to establish any sort of context from within the host or system that is generating the event”).

25. Figure 1 below illustrates a known network forensics system utilizing a network edge device to monitor, capture, and analyze network traffic between the network edge device 116 and the computing devices 104 and 108:

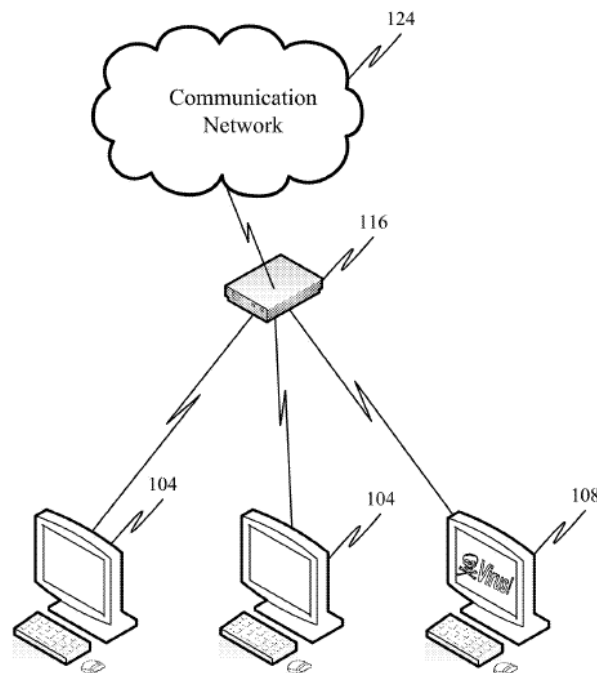


Figure 1

Id., Fig. 1. Network edge device 116, which may be referred to as a “sniffer,” “may monitor, capture, and analyze network traffic between the computing devices 104 and 108 and between the computing devices 104/108 and the communication network 124.” *Id.*, 1:50-62. However, this sniffer “as no ability to monitor, capture,

and/or analyze what is occurring within each of the computing device 104 and 108.” *Id.*, 1:63-66; *see also id.*, 2:10-12 (noting this sniffer “cannot provide details regarding files modified, registry entries changed, and/or new processes that are created” within computing devices 104/108 and “has no way of knowing to what extent the virus may have impacted the computing device 108, what additional files may have become infected, or what processes and/or objects operating are linked to the virus”)

26. The '045 Patent claims to have improved network forensics by providing methods by which “events occurring within a computing device are captured and additional context and a global perspective is provided for each capture event.” *Id.*, 2:23-28. These methods include, for example:

gathering one or more events defining an action of a first object acting on a target; generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object; obtaining a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware; and Internet Protocol (IP) Address or Uniform Resource Locator (URL) information; assembling an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmitting the assembled event line.

Id., 4:44-63.

27. In more detail, the '045 Patent's methods are broken down as follows: (1), "system filters may intercept system events" containing three main components: "information and/or details describing the object performing the act (the 'Actor'), information and/or details describing the act being performed (the 'Event Type'), and information and/or details describing the object or identity of another entity upon which the act is being performed (the 'Victim' or 'Target')[;]" (2), contextual state information is generated "by correlating events to objects and creating an audit trail for each individual event," where such "[c]ontextual state information may include...originating processes, devices, and users[;]" and additionally the "GenesisActor," which is the "ultimate parent of the event being performed[;]" (3), a global perspective is obtained which may include "information related to the age, popularity, and determination of an object of the event—whether it is known good, bad, or unknown[;]" (4), an event line is created "comprising information from the event[;]" and (5) the event line is transmitted. *See generally, id.*, 7:29-10:2. In this way, an "audit trail" is created which allows an event to be "forensically track[ed] back" to what triggered it. *Id.*, 12:59-63.

B. Overview of the '045 Patent's File History

28. I understand that the application which issued as the '045 Patent—Application No. 14/270,069 (the "'069 Appl.")—was filed on May 5, 2014 and

claims priority to a provisional application filed on May 3, 2013. I have been informed that this means the first inventor to file provisions of the Leahy-Smith America Invents Act (AIA) apply to Petitioner's challenge and my opinions herein. I apply this May 3, 2013 date for the purposes of my opinions herein only as the priority date for the Challenged Claims.

29. I understand the following limitation was the sole added addition to the original claims of the '069 Appl. which resulted in the allowance of the '045 Patent: "wherein the global perspective for one or more related events to the at least one event across a network." However, for the reasons discussed below, it is my opinion that this limitation has long been found in the art and is thus not "new" as viewed by one of skill in the art.

30. On September 16, 2015, the Examiner rejected claims 1-19 as they existed then as unpatentable under 35 U.S.C. 102(a)(2) over U.S. Publication No. 2013/0298244 to Kumar et al. ("*Kumar*"). '045 *Patent File History* (Ex. 1002), 98-102. I have analyzed *Kumar* (Ex. 1004), and this reference generally relates to "[s]ystems and methods for threat identification and remediation for computing platforms based upon reconnaissance-based intelligence correlation and network/application monitoring" which includes a "security orchestration service generates runtime operational integrity profiles representing and identifying a level of threat or contextual trustworthiness, at near real time, of subjects and applications

on the instrumented target platform.” *Kumar*, Abstract. More particularly, as Applicant admitted during prosecution, *Kumar* relates to:

methods for threat identification and remediation. *Kumar* teaches a security orchestration service that generates runtime operational integrity profiles representing and identifying a level of threat or contextual trustworthiness of subjects and applications on the instrumented target platform. *Kumar* also teaches a method for threat identification and remediation for computing platforms based upon reconnaissance-based intelligence correlation and network/application monitoring are disclosed. The method comprises receiving: a dynamic context including endpoint events and network endpoint assessments; generating temporal events based on the network endpoint assessments; and correlating the endpoint events and temporal events before generating an integrity profile for the system.

'045 Patent File History, 83-84.

31. In Applicant’s March 16, 2016 Response to the September 16, 2015 Office Action, Applicant argued that *Kumar*’s disclosures were deficient regarding two limitations: teach the “contextual state” and “global perspective” limitations. *Id.*, 84-85. On May 31, 2016, the Examiner rejected Applicant’s arguments concerning *Kumar* and Applicant subsequently filed a substantive amendment to the claims adding in the “wherein the global perspective for one or more related events to the at least one event across a network.” *Id.*, 42-43. Applicant argued *Kumar* failed to teach the newly added limitation because *Kumar*’s GUI “presents information related to the integrity of a single application running on a single device.” *Id.*, 52. Applicant argued that “[t]here is no information displayed about events related to

the application integrity across other devices on a network... [such that] Kumar cannot possibly teach or suggest a ‘global perspective for one or more related events to the at least one event across a network[.]’” *Id.* A Notice of Allowance issued on October 5, 2016. *Id.*, 22-25.

C. Level of a Person Having Ordinary Skill in the Art

32. I have been asked to provide my opinion as to the level of skill of a person having ordinary skill in the art at the time of the ’045 Patent, which I have been instructed to assume is May 3, 2013. In determining the characteristics of a hypothetical person having ordinary skill in the art at the time of the claimed invention, I was told to consider several factors, including the type of problems encountered in the art, the solutions to those problems, the rapidity with which innovations are made in the field, the sophistication of the technology, and the education level of active workers in the field. I also placed myself back in the time frame of the claimed invention and considered the colleagues with whom I had worked at that time.

33. In my opinion, a person having ordinary skill in the art, as of May 3, 2013, would have been a person having a bachelor’s degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and would have had a working knowledge of host monitoring systems, software security analysis, and dynamic malware analysis. Additional graduate education

could substitute for professional experience, or significant experience in the field could substitute for formal education. Such a person would have been capable of understanding the '045 Patent and the prior art references discussed below.

34. I was asked whether less experience would suffice here as compared to the amount of experience required for the technology at issue in the proceeding involving U.S. Patent No. 10,284,591, where I similarly offered opinions. As stated in my opinions therein, the '591 Patent generally relates to the intricacies of anti-exploit systems which employ stack walk processing to detect suspicious behavior and prevent exploit code from being executed. The granularity of detail involved in the specific anti-exploit techniques requires slightly more academic or industry experience than the malware detection techniques described by the '045 Patent. While the technology at issue in the '045 Patent is still complex and four or more years of industry experience would be preferred, it is my opinion that a person having a bachelor's degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and a working knowledge of host monitoring systems, software security analysis, and dynamic malware analysis will suffice here and would be capable of understanding the '045 Patent and the prior art at issue.

35. Based on my education, training, and professional experience in the field of the claimed invention, I am familiar with the level and abilities of a person

of ordinary skill in the art at the time of the claimed invention. Additionally, I met at least these minimum qualifications to be a person having ordinary skill in the art as of the time of the claimed invention of the '045 Patent.

D. Background of the Technology

36. I was asked to briefly summarize the background of the prior art from the standpoint of a person having ordinary skill in the art prior to May 3, 2013. As explained below, the forensic computer security techniques described by the '045 Patent were well-known and conventional prior to the invention of the '045 Patent.

37. An increasing number of computers are connected to computer networks, such as the Internet. Networked computers provide a significant benefit of accessing and sharing data over networks. However, these networked computers are also vulnerable to attacks, unwanted intrusions, and unauthorized access. Network forensics was established to combat these vulnerabilities. “Network forensics deals with the capture, recording or analysis of network events in order to discover evidential information about the source of security attacks[.]” *Tools and Techniques for Network Forensics* by Meghanathan et al. (“*Meghanathan*”) (Ex. 1012), 1.

38. Several different tools and techniques were available to conduct network forensics. For example, some common network forensic tools include the following: “eMailTrackerPro – to identify the physical location of an email sender;

Web Historian – to find the duration of each visit and the files uploaded and downloaded from the visited website; packet sniffers like Ethereal – to capture and analyze the data exchanged among the different computers in the network.”
Meghanathan, 1.

39. While these tools in isolation were helpful at identifying rudimentary attacks, the industry developed a need for more advanced and integrated security measures to better protect their systems. Both network-based intrusion and host-based detection systems were developed to compliment these tools. *Host- vs. Network-Based Intrusion Detection Systems* (Ex. 1013), 2. “Intrusion detection systems are security systems that collect information from various types of system and network sources, and analyzes this data in an attempt to detect activity that may constitute an attack or intrusion on the system.” *Id.*; *see also id.*, 2-3 (noting IDS’s provided the ability for “system managers to more easily handle the monitoring, audit, and assessment of their systems and networks, which is a necessary part of security management”).

40. Regarding network-based intrusion detection systems, these systems operate off a “wiretapping concept,” meaning information is collected from the network itself over a network traffic stream, typically in the form of network packets. *Id.*, 5. An analysis is conducted on these “tapped” packets, such that the contents and header information are inspected and compared to attack signatures to identify an

attack. *Id.* While network IDS's monitor traffic traveling over a network, these systems "have a degree of blindness to host activity." *Id.*; see also *id.*, 5-6 ("Although some network-based systems can infer from network traffic what is happening on hosts, they cannot tell the outcomes of commands executed on the host"). This can lead to false positives for attack detections and wasted resources. *Id.*, 6.

41. To analyze events occurring on a host, host-based intrusion detection systems were employed. "Host-based intrusion detection systems are aimed at collecting information about activity on a particular single system, or host." *Id.*, 3. In a typical host-based IDS, a sensor agent is installed on a host which "collect[s] data about events taking place on the system being monitored." *Id.* "This data is recorded by operating system mechanisms called audit trails[.]" *Id.* This audit trail information "allows the intrusion detection system to spot subtle patterns of misuse that would not be visible at a higher level of abstraction." *Id.* (noting "[t]he information collected through audit trails can arm the host-based sensor with useful data about the system and its users"). For example, "audit trails may contain information about subjects responsible for an event, as well as any objects related to that event." *Id.*, 3-4. "The host-based sensor can recover which process initiated an event, and the current and original user identifications associated with that event, in case the user identification changes." *Id.*, 4. This data is "crucial in determining from

what program and by what user a potential network attack originated, which will obviously help in stopping future attacks.” *Id.* These host-based systems are desirable because they can “monitor access to information in terms of ‘who accessed what,’” such that “these systems can trace malicious or improper activities to a specific user ID.” *Id.*, 4.

42. However, without sufficient contextual information concerning an attack, these host-detection systems were similarly prone to false positives and adversely disrupted technology operations. For instance, *Evolving Endpoint Malware Detection: Dealing with Advanced and Targeted Attacks* by Securosis (Ex. 1014), recognizes this principle. Because attackers continue to advance their tactics, “efforts to detect malware can no longer focus exclusively on what malware looks like (a file hash or some other identifying factor) and must incorporate a number of new data sources for identification.” *Id.*, 7. These new sources include “what it does, how it gets there, and who sent it; in combination with traditional file analysis[.]” *Id.* This “broader information base enables you to improve accuracy and reduce false positives.” *Id.* More particularly, for “accurate and effective detection you need better context on what the code does, where it came from, and who it came from, in order to reach a reasonable verdict on whether to allow or block execution.” *Id.*, 11.

43. A system combining both traditional file analysis in conjunction with additional contextual information is described in *Combining File Content and File*

Relations for Cloud Based Malware Detection by Yanfang Ye et al. (“Ye”) (Ex. 1015). *Ye* describes a cloud-based malware detection system adopted by Comodo Security Solutions Inc. *Id.*, 2. The workflow of this system is described below in Figure 2:

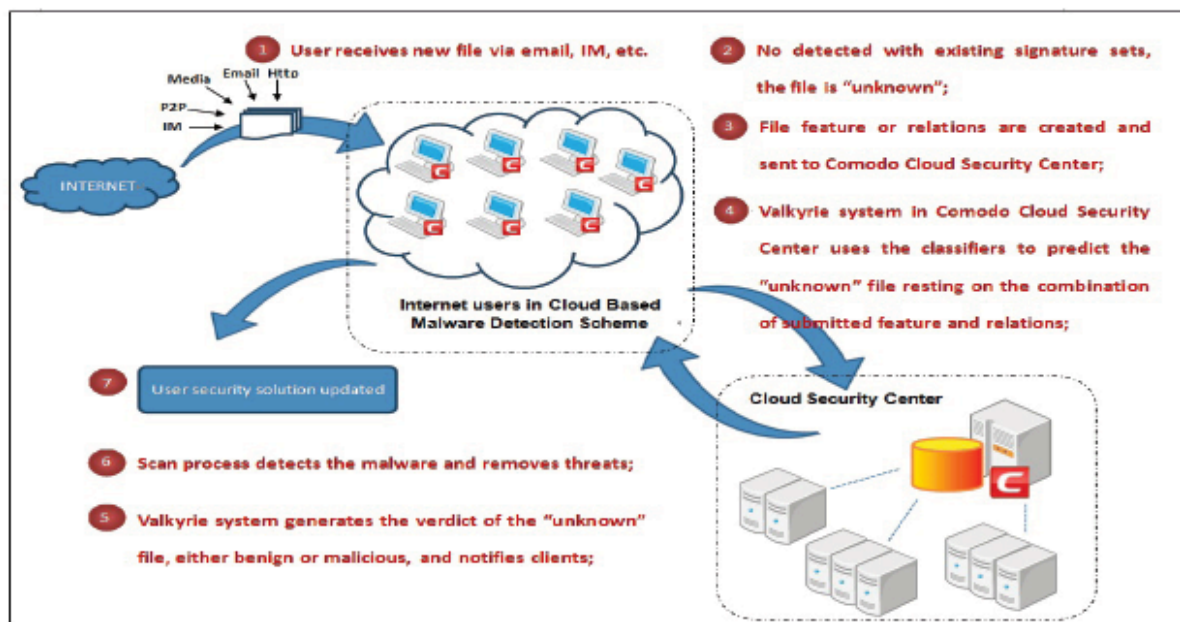


Figure 2: The Workflow of Comodo Cloud Based Malware Detection Scheme.

Id. This workflow is explained in more as follows:

1. On the client side, users may receive new files from emails, media or IM (Instant Message) tools when they are using the Internet.
2. Anti-malware products will first use the signature set on the clients for scanning. If these new files are not detected by existing signatures, then they will be marked as “unknown”.
3. In order to detect malware from the unknown file collection, file features (like file content as well as the file relations) are extracted and sent to Comodo Cloud Security Center.

4. Based on these collected features, the classifier(s) on the cloud server will predict and generate the verdicts for the unknown file samples, either benign or malicious.
5. Then the cloud server will send the verdict results to the clients and notify the clients immediately.
6. According to the response from the cloud server, the scanning process can detect new malware samples and remove the threats.
7. Due to the fast response from the cloud server, the client users can have most up-to-date security solutions.

Id.

44. With respect to step 3 above, *Ye*'s system extracts both file content and file relations to send to the backend cloud security center. *Ye* explains "the relations among file samples provide invaluable information about their properties." *Id.*, 3. "Relation information provides a graph structure in the data and induces pairwise similarity between objects while the file content provides inherent characteristic information about the file samples." *Id.*, 3-4. Utilizing both sets of information provides better performance for malware detection systems. *Id.*, 4.

45. Example file "relation" information is shown below in Figure 3



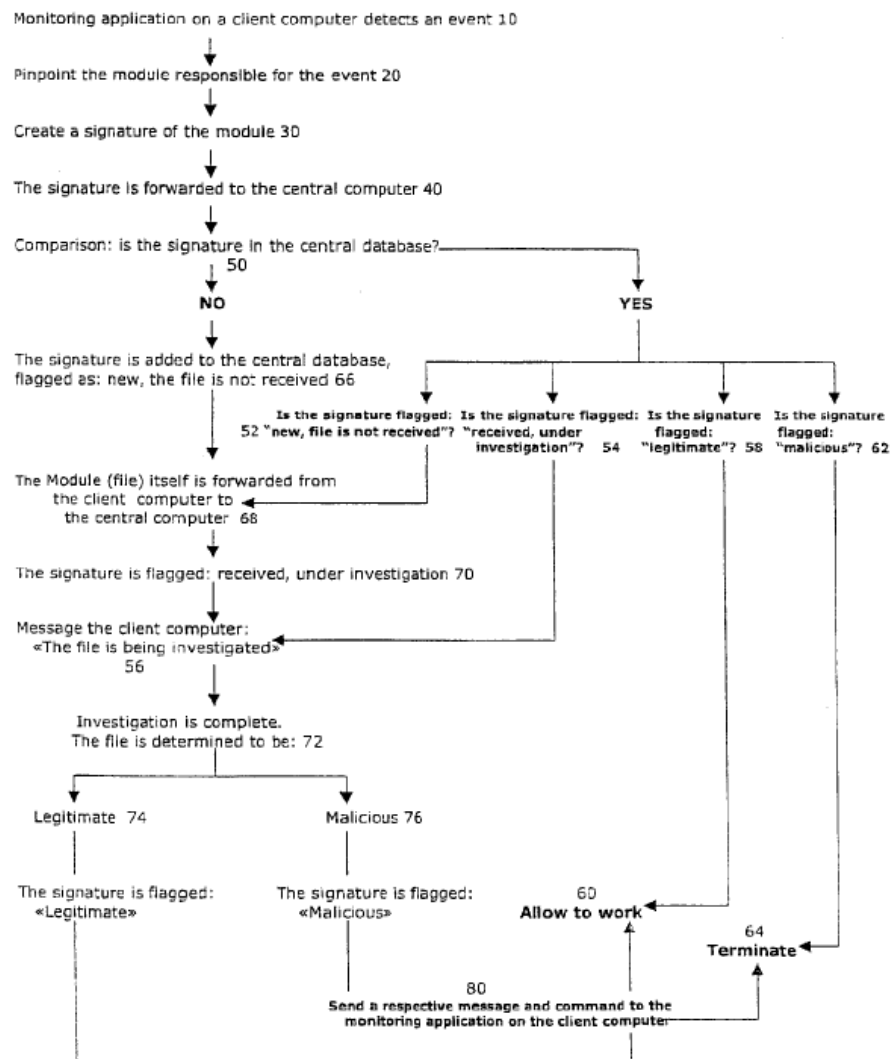
Id., 3. Here, we can observe the file “yy(1).exe” is associated with many trojans marked in purple. From the relations shown in Figure 3, we can infer that if an unknown file always co-occurs with numerous kinds of trojans in users’ computers, then most likely, this unknown file is a malicious Trojan-Downloader file. *Id.* To confirm this, additional file content information can be analyzed, such as API calls from the collected files, and this can be sent to a classification model on the cloud server for interpretation. *See generally, id.*, 4-7.

46. *Ye's* cloud-based malware detection system is a subtype of a larger umbrella of “community-based” anti-malware systems. In a community-based anti-malware system, a plurality of user computers are all connected via a network (e.g., the Internet) to a central computer. When a new file is encountered, a user computer can send the file or audit information concerning the file to the central computer for

analysis. This information can be used to make comparisons with this data across the community to determine whether the file is malicious or not.

47. An example community-based malware detection system is found in U.S. Patent No. 6,944,772 to Dozortsev (“*Dozortsev*”) (Ex. 1016). A working session of *Dozortsev*’s system is depicted below in Figure 1:

Fig. 1.

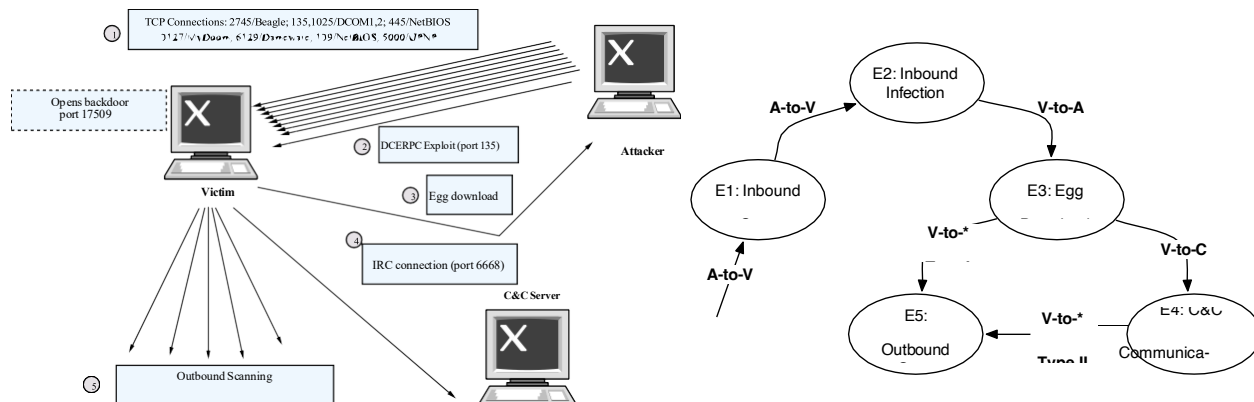


Id., Fig. 1. As shown above, monitoring software detects an event 10 and attempts to pinpoint executable code responsible for the event. *Id.*, 7:18-33. A signature is created for this code using MD5 or another algorithm, and this signature is forwarded to a central computer 40 for analysis. *Id.*, 7:33-37. “The first step of analysis of the suspect signature includes comparison of the suspect signature with the plurality of signatures stored in the database 50.” *Id.*, 7:38-40. In this way, the system can compare this signature with data across the community to determine if the code is malicious. *Id.*, 7:40-8:41 (noting this analysis can be in a “sandbox environment” to safely execute this code to make a malware determination). If malicious, the monitoring application receives a command to terminate the event. *Id.*

48. Another example of a community-based malware detection system is found in U.S. Patent No. 6,772,363 to Chess et al. (“*Chess*”) (Ex. 1017). Here, a set of client computers are connected over a network to a central analysis center whereby files which are suspected of containing viruses, Trojan horses, worms, or other types of malicious code are passed to the central analysis center for comparison with other known files across the community. *Id.*, 1:66-2:28, 5:57-6:21. The central analysis center performs analysis, for example in a simulated decoy environment, and determines if the file contains malicious code. *Id.*, 6:45-66; *see also* WO 2002/033525 to Shyne-Song Chuang (Ex. 1018), EP 1,549,012 to Kristof De

Spiegeleer (Ex. 1019), and EP 1,280,040 to Alexander James Hinchliffe, et al. (Ex. 1020) (all describing further community-based malware detection systems).

49. Many commercial systems were in use, aimed at combatting more advanced instances of malware. For example, sophisticated malware infections such as botnets became prevalent in the early 2000s'. These infections typically involved multiple malware sources and command centers, multiple incoming and outgoing attack activities, and longer time duration. In response, the research community and industry developed multi-step detection and correlation analysis approaches. For example, Gu et al. developed the BotHunter system in 2007. *See* Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee., *BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation*, In Proceedings of the 16th USENIX Security Symposium August 2007 (Ex. 1026). BotHunter consists of malware-focused sensors to detect specific stages of the malware infection, “including inbound scanning, exploit usage, egg downloading, outbound bot coordination dialog, and outbound attack propagation.” *Id.*, 1. This system then uses a correlation engine to “tie together the dialog trail of inbound intrusion alarms with those outbound communication patterns that are highly indicative of” (*id.*, 1) a bot malware infection that corresponds to an infection dialog model. *Id.*, 4. This is shown in the figures below:



50. The research community and industry had also developed analysis tools to extract the not-yet-in-use malware behaviors that can be activated in the future. Malware detection systems can then be equipped ahead of time with the corresponding detection models. For example, Nadji et al. developed a system called GZA in 2011 to play network games with malware and force it to reveal the alternative, or, backup, malware servers on the Internet. See Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee, *Understanding the Prevalence and Use of Alternative Plans in Malware with Network Games*, In Proceedings of The 27th Annual Computer Security Applications Conference, December 2011 (Ex. 1027). When running malware in a sandbox, GZA fools the malware that its current server is no longer available and the malware is forced to try to connect to a backup server, hence revealing its identity.

51. It is well-known that the security industry is a community that shares attack information including malware infection data such as source, target, severity, etc. Rather than including only the information pertinent to a specific malware or an actual attack incident, the reputation, or, the likelihood of an entity such as a new website can also be used for security analysis. For example, Antonakakis et al. developed a DNS reputation system called Notos in 2010. *See* Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster, *Building a Dynamic Reputation System for DNS*, In Proceedings of The 19th USENIX Security Symposium, August 2010 (Ex. 1028). Notos “builds models of known legitimate domains and malicious domains, and uses these models to compute a reputation score for a new domain indicative of whether the domain is malicious or legitimate.” *Id.*, 1. Notos “uses information about malicious domain names and IP addresses obtained from sources such as spam-traps, honeynets, and malware analysis services to build a model of how network resources are typically allocated by Internet miscreants. With these models, Notos can assign reputation scores to new, previously unseen domain names, therefore enabling dynamic blacklisting of unknown malicious domain names and IP addresses.” *Id.*, 2. The intuition behind Notos is guilty by association. For example, if a new domain is hosted on a machine that has in the past hosted other domains that are known to have been used for

malware download and command and control, then the new domain is likely to be also for malicious use.

52. Furthermore, user behaviors can be monitored to identify not only the source of a downloaded file but also the user's consent for downloading the file. For example, Lu et al. developed a system called BLADE drive-by malware infections in 2010. See Long Lu, Vinod Yegneswaran, Phil Porras, and Wenke Lee., *BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections*, In Proceedings of The 17th ACM Conference on Computer and Communications Security (CCS), October 2010 (Ex. 1029). BLADE uses "user-interaction tracking as a means to collect user download authorizations expressed during interactions with browsers. BLADE captures on-screen consent-to-download dialog windows via a Screen Parser module and tracks the user's physical interactions (e.g., mouse clicks) with these dialog windows via a Hardware Event Tracer." *Id.*, 1. BLADE also uses consent correlation to "discern 'transparent' downloads from those that involve direct user authorization." *Id.*, 1. BLADE also uses "disk I/O redirection to efficiently contain disk footprints of unconsented data delivered through supervised processes." *Id.*, 1. Further, "[f]ile content created and manipulated by supervised processes is stored in a nonexecutable location created and managed by BLADE." *Id.*, 2. And "[f]iles in the secure zone are prevented from being loaded into the memory as executables (in the case of Windows, load operations of .exe, .dll, .sys,

and etc. are disallowed).” *Id.*, 2. BLADE’s architecture is illustrated in the figure below:

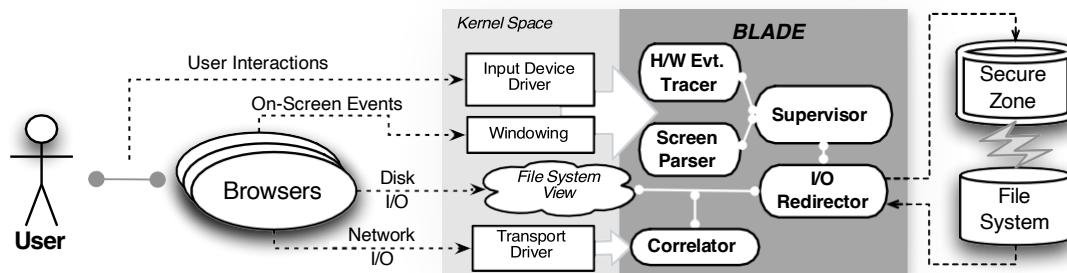


Figure 1: Overview of the BLADE system architecture

E. Claim Construction

53. I have been informed by counsel that the first step in a patentability analysis involves construing the claims, as necessary, to determine their scope. Second, the construed claim language is then compared to the disclosures of the prior art. I am informed that claims are generally given their ordinary and customary meaning as understood by a person having ordinary skill in the art at the time of the invention, in light of the patent specification. I have applied the plain and ordinary meaning to the claim terms at issue herein unless otherwise noted. *See, e.g.*, Sections III.G.1.f and III.H.1.f.

F. Summary of the Prior Art References

1. Morris

54. *Morris* (Ex. 1005) discloses a “method of classifying a computer object as malware” including “receiving at a base computer data about a computer object

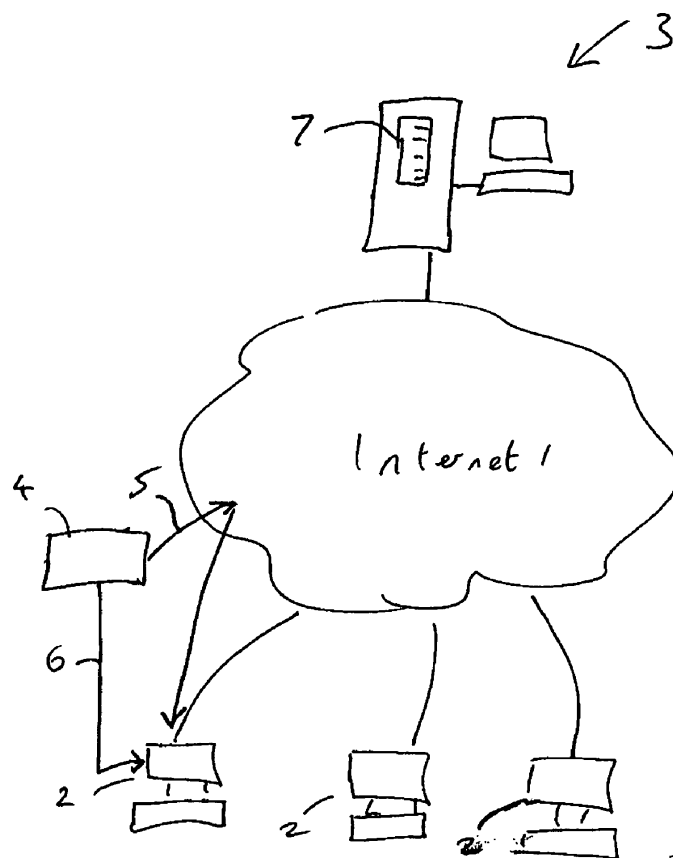
from each of plural remote computers on which the object or similar objects are stored...[t]he data about the computer object received from the plural computers is compared in the base computer” and the “computer object is classified as malware on the basis of said comparison.” *Morris*, Abstract. “[T]he data about the computer object includes...events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.” *Id.*

55. *Morris* explains that “[c]ompared to the prior art that relies solely on signature matching,” its methods allow “a comparison to be made between the objects and/or their effects on the different remote computers to determine whether or not a particular object should be classed as good or as malware.” *Id.*, [0017]. “This allows a rapid determination of the nature of the object to be made, without requiring detailed analysis of the object itself as such to determine whether it [is] malware and also avoids the need to generate new signatures to be used for signature matching as in the conventional prior art anti-virus software.” *Id.*

56. *Morris* discloses this comparison between objects can include identifying “relationships between the object and other objects” such that the system can “immediately and automatically to mark a child object as bad (or good) if the or a parent or other related object is bad (or good).” *Id.*, [0019]. “[I]f at least one other object to which said object is related is classed as malware, then the method may comprise classifying said object as malware.” *Id.* “Said other objects include the

object or similar objects stored on at least some of the remote computers.” *Id.* “Said other objects may include other objects that are parent objects or child objects or otherwise process-related objects to said object.” *Morris*’s methods “may be carried out by so-called agent software running on the remote computer.” *Id.*, [0033].

57. *Morris*’s methods can be conducted by the following distributed architecture depicted below in Figure 1:



Id., Fig. 1. Here, “[p]lural local or ‘remote’ computers 2” (e.g., personal computer, server of any type, a PDA, mobile phone, etc.) “are connected via the Internet 1 to a ‘central’ or ‘base’ computer 3.” *Id.*, [0077]. An object 4 may be “downloaded to a remote computer 2 via the Internet 1 as shown by lines 5 or applied directly as shown

by line 6.” *Id.* (noting an object may be a “computer file, part of a file or a sub-program, macro, web page or any other piece of code to be operated by or on the computer, or any other event whether executed, emulated, simulated or interpreted”). Object 4 is run at a remote computer 2 and a local database at remote computer 2 is searched to identify “a signature or key representing that particular process, its related objects and/or the event.” *Id.*, [0079]. “If the object is unknown locally, then details of the object are passed over the Internet 1 or other network to the base computer 3 for storing in the community database 7 and preferably for further analysis at the base computer 3.” *Id.*, [0081]. “The details of an object 4 that are passed to the base computer 3 are preferably in the form of a signature or ‘key’ that uniquely identifies the object 4.” *Id.*, [0086].

58. These details of the object may include “event” metadata, which “define the actions or behaviours of an object acting upon another object or some other entity.” *Id.*, [0105]. “The event has three principal components: the key of the object performing the act (the ‘Actor’), the act being performed (the ‘Event Type’), and the key of the object or identity of an other entity upon which the act is being performed (the ‘Victim’).” *Id.* Examples of these events include the following:

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. “Genesisactor” metadata can also be captured. *Id.*, [0107]. The Genesisactor is the “key of an object that is not the direct Actor of an event but which is the ultimate parent of the event being performed.” *Id.*

59. As stated above, “further analysis” is conducted on this data at base computer 3, including “central heuristics” which “use five distinct processes to establish if an object is safe, unsafe or suspicious.” *Id.*, [0097]; *see generally, id.*, [0098]-[0103] (describing these five heuristics). These heuristics are “run on the data in the community database 7.” *Id.*, [0095]. For example, the second heuristic “utilises an object’s relationship to other objects that act upon it or upon which it acts” to create a “simple 1-to-1 relationship chain” which “provides a complex series of correlation points, allowing ancestral relationships to be considered for any object and its predecessors by event or its issue (i.e. child and sub-child processes) by event.” *Id.*, [0099].

60. As I discussed above, I have been informed that for a prior art reference to be proper for use in an obviousness analysis, the reference must be “analogous art” to the claimed invention. I have been informed that a prior art reference is analogous to the claimed invention if, for example, the reference is from the same field of endeavor as the claimed invention or is reasonably pertinent to the problem faced by the inventor.

61. The '045 Patent is directed generally to methods for providing computer security by analyzing event data from potential threat events. For example, the '045 Patent describes methods for determining whether objects are malware, comprising:

gathering one or more events defining an action of a first object acting on a target; generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object; obtaining a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware; and Internet Protocol (IP) Address or Uniform Resource Locator (URL) information; assembling an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmitting the assembled event line.

'045 Patent, 4:42-62; *see also id.*, 4:63-5:19, 9:39-49, 23:4-13, claims 1, 8, 17, and 19. As described above, *Morris* is similarly directed to methods for providing computer security by analyzing event data from potential threat events, including determining whether objects are malware. *See, e.g., Morris*, [0001] (“The present invention relates generally to methods and apparatus for dealing with malware”), [0014]-[0017] (method of classifying a computer object as malware), [0019] (“if at least one other object to which said object is related is classed as malware, then the

method may comprise classifying said object as malware”), [0098]-[0100] (describing using object relationships from an event “for the rapid identification and determination of malware”), [0128]-[0134]. *Morris* is therefore in the same field of endeavor as the ’045 Patent.

62. The ’045 Patent is also directed at a problem of using ancestral or audit information to make a malware determination. ’045 Patent, 3:45-47 (“The analysis starts by correlating events to objects (their originating processes, devices, and users), and creating an audit trail for each individual event”), 8:30-34 (“Following the aggregation and interpretation of the events, the context analyzer creates, or obtains, a contextual state by correlating events to objects and creating an audit trail for each individual event”), 12:57-60 (“For example, the context analyzer 348 may correlate an actor, victim, and/or event type to one or more originating processes, devices, and users. Such a correlation creates an audit trail for each individual event”), 18:55-57 (“context analyzer correlates events to objects such that an audit trail for each individual event is created”), claims 1, 2, 8, 10, 16, 17, 19 (all claiming an “originating object” associated with an event). Likewise, as described above, *Morris* is directed using ancestral or audit information to make a malware determination. *See, e.g., Morris*, [0052] (“uses the concept of ancestry to enable objects to be marked as malware...[f]or example, any particular process may spawn child processes which are therefore related...[t]he key relating to the first object may

be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object”), [0099] (“This simple 1-to-1 relationship chain provides a complex series of correlation points, allowing ancestral relationships to be considered for any object and its prede-cessors by event or its issue (i.e. child and sub-child pro-cesses) by event”), [0128]-0135] (describing “gene” analysis using the concept of ancestry to enable objects to be marked as malware). *Morris* is therefore reasonably pertinent to a problem faced by the inventor.

2. *Van Oorschot*

63. *Van Oorschot* (Ex. 1006) is directed to “a method of managing system events in a network.” *Van Oorschot*, 1:55-2:15. In this method, a “computer receives a system event initiated by an initiating client associated with a user[,]” “[t]he system event comprises a plurality of data elements associated with respective ones of a plurality of system event attributes[,]” and then this system event is compared against a security policy to determine a severity level for the event. *Id.* To accomplish this method, *Van Oorschot* discloses an “event management tool” operable to “record, manage, and respond to, system events that occur on the local network.” *Id.*, 3:29-31.

64. For example, this method can be conducted on the distributed network depicted below in Figure 1:

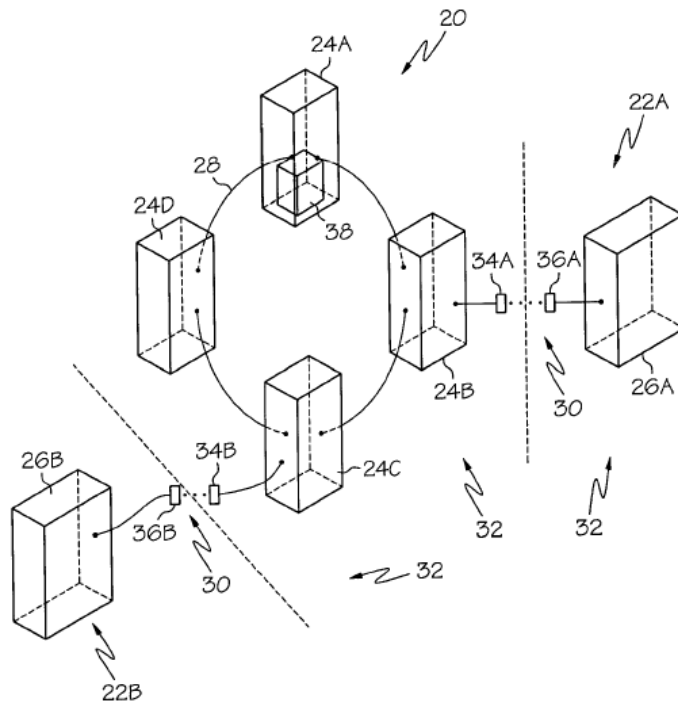


FIG. 1

Id., Fig. 1. Here, a “system event is monitored by the operating system of one of the local servers 24A-D, for example server 24D, and is logged in a suitable event log, along with multiple other system events, according to operating system convention of the host of the event.” *Id.*, 3:28-36. “The event management tool software or process 38, active on server 24A, periodically polls, or otherwise monitors, the system event logs of other servers 24B-D in the network, as well as the system event logs of its own host server 24A, and collects information stored in the system event logs.” *Id.*, 3:37-41; *see also id.*, 3:49-56 (noting the event management tool may also “process system events in real time, as they are generated, or may optionally obtain information related to system events from applications running on the operating

system”). The event management tool can catalogue and manage these system events to “provide for a comprehensive security overview of activity in the local network 20” such that a “forensic statement of the activity is generated” and security alerts can be issued. *Id.*, 3:57-4:3.

65. An original event log is depicted below in Figure 5A:

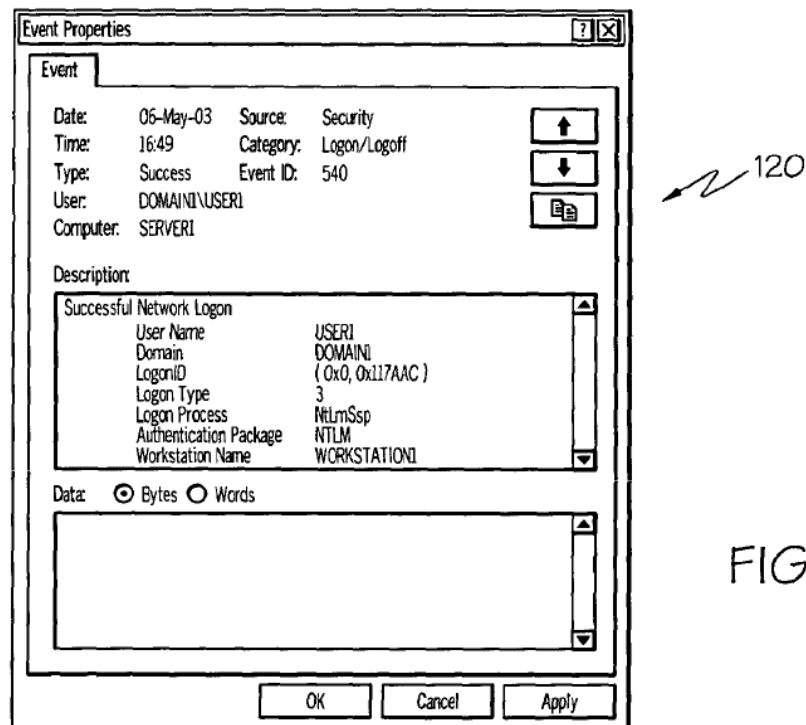


FIG. 5A

Id., Fig. 5A. “The event log 120 of FIG. 5A is the result of a user USER1 logging onto a Windows/NT 2000™ fileserver from his own workstation WORKSTATION1” and “[t]he event occurs on a machine SERVER1, i.e. the ‘computer field’ in the original event 120.” *Id.*, 6:4-14. “The event was initiated from a machine WORK-STATION1, i.e. the ‘workstation name’ in the original event

120.” *Id.* The event management tool then normalizes this data, capturing the following attributes:

“‘who’ (the client initiating the event); ‘what’ (the type of event that was initiated); ‘when’ (the time at which the event occurred); ‘where’ (the system or device on which the event occurred); and ‘on what’ (the file, device, or setting that was accessed by the event)...[and] ‘where from’ (the identity of the client initiating the event) and ‘where to’ (the identity of the client acted upon by the event)”

Id., 6:14-23, Fig. 5B; *see also id.*, Figs. 6A and 6B (depicting an example of an original event log and a normalized event log for a system event of a Checkpoint Firewall-I Firewall™ server). A “severity level” is then assigned to the event, which “provides an additional gauge by which a decision can be made as to the manner in which an event is to be processed or as to the relative importance of the event.” *Id.*, 8:4-11, Fig. 11.

66. As I described above, the ’045 Patent is directed generally to methods for providing computer security by analyzing event data from potential threat events. *See above*, ¶¶24-27 (citing ’045 Patent at 4:42-62, 4:63-5:19, 9:39-49, 23:4-13, claims 1, 8, 17, and 19). As I explained in the preceding paragraphs, *Van Oorschot* is similarly directed to methods for providing computer security by analyzing event data from potential threat events. *See, e.g., Van Oorschot*, 1:37-42 (describing analyzing security events and generating a “forensic statement” to alert for security issues), 3:63-4:3 (describing providing a “comprehensive security overview of activity on the local network” and generating a “forensic statement” of the activity”

to generate security alerts and reports); *see also id.*, 1:55-2:15 (describing method for capturing event data and generating security alerts). *Van Oorschot* is therefore in the same field of endeavor as the '045 Patent.

67. The '045 Patent is also directed at a problem of providing rules for handling security events. *See*, '045 Patent, 9:28-49 (describing “rule set” which “may determine that an event meeting certain conditions be assigned a priority according to a scale of severity”), 13:34-46 (describing the same), claim 3. Likewise, *Van Oorschot* is directed at providing “policy rules” which can identify an event as a threat and indicate a “severity level” for the threat. *Van Oorschot*, 7:34-57; *see also id.*, 8:4-11 (“severity level” “provides an additional gauge by which a decision can be made as to the manner in which an event is to be processed or as to the relative importance of the event”), 9:4-17 (severity level based on special attention rule). *Van Oorschot* is therefore reasonably pertinent to a problem faced by the inventor.

3. Capalik

68. *Capalik* (Ex. 1008) is directed to “systems and methods for protecting computer networks, including but not limited to systems and methods for analyzing malicious activities on a computer system in order to better protect the computer system from future malicious activity.” *Capalik*, [0004]. *Capalik* recognizes an “increasing number of computers are connected to computer networks (e.g., the Internet)” and through such connections, these networked computers are “vulnerable

to attacks, unwanted intrusions, and unauthorized access.” *Id.*, [0005]. Existing network security systems (e.g., firewall, IDS, IPS) provide some protection from these malicious activities, but often lack sufficient security for so-called “zero-day attacks”—meaning “security vulnerabilities previously unknown to software developers or system operators.” *Id.*, [0006]-[0008]. *Capalik*, therefore, provides “computer-implemented methods and systems for generating fingerprint data for zero-day unauthorized activities.” *Id.*, [0009].

69. *Capalik’s* computer-implemented methods are “performed at a computer system having one or more processors and memory storing one or more programs for execution by the one or more processors.” *Id.*, [0010]. This system generally includes the following:

In accordance with some embodiments, the system includes one or more processors, and memory. The memory stores a virtual machine, a virtual machine monitor supervising the virtual machine, and one or more programs. The virtual machine, the virtual machine monitor, and the one or more programs are configured for execution by the one or more processors. The one or more programs include instructions for monitoring activity on the virtual machine, and instructions for identifying a plurality of activities being performed at the virtual machine. Each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target. The one or more programs furthermore include instructions for storing the activity sources, activity targets, and associations in the memory. The one or more programs also include instructions for creating a fingerprint indicative of the activity on the virtual machine from the stored activities. Finally, the one or more programs include instructions for transmitting the fingerprint to one or more other computer systems on the network to prevent future attacks that comprise the same or similar activities as

indicated by the fingerprint. In some embodiments, the one or more programs include instructions for displaying at least a subset of the activities (e.g., action sources, action targets, and actions).

Id., [0011].

70. For example, Figure 1 below “illustrates an exemplary distributed computer system 100” which effectuates *Capalik’s* methods:

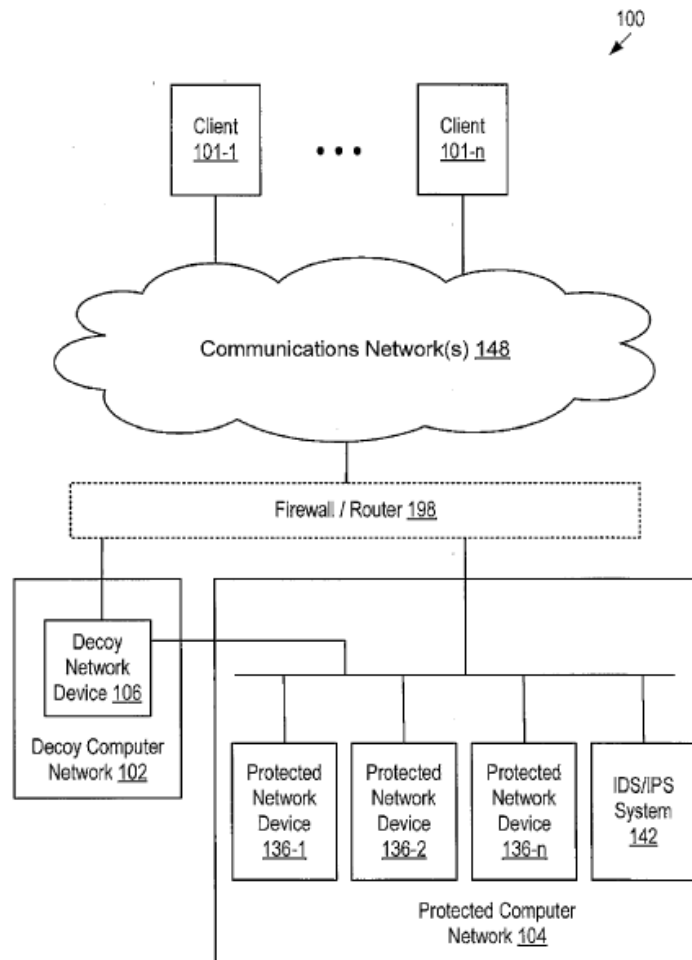


Figure 1

Id., Fig. 1. As shown above, system 100 includes a decoy computer network 102 comprising a decoy network device 106. *Id.*, [0025]. “The purpose of the decoy

network device 106 is to allow attackers to attack the decoy network device 106, so that the pattern of the attack can be monitored and analyzed to generate a fingerprint.” *Id.*, [0030]. “This fingerprint of the attack can be used to prevent similar attacks on the decoy network device 106 and/or other computers (e.g., protected network devices 136).” *Id.*

71. The components of *Capalik’s* decoy network device 106 are depicted below in Figure 2:

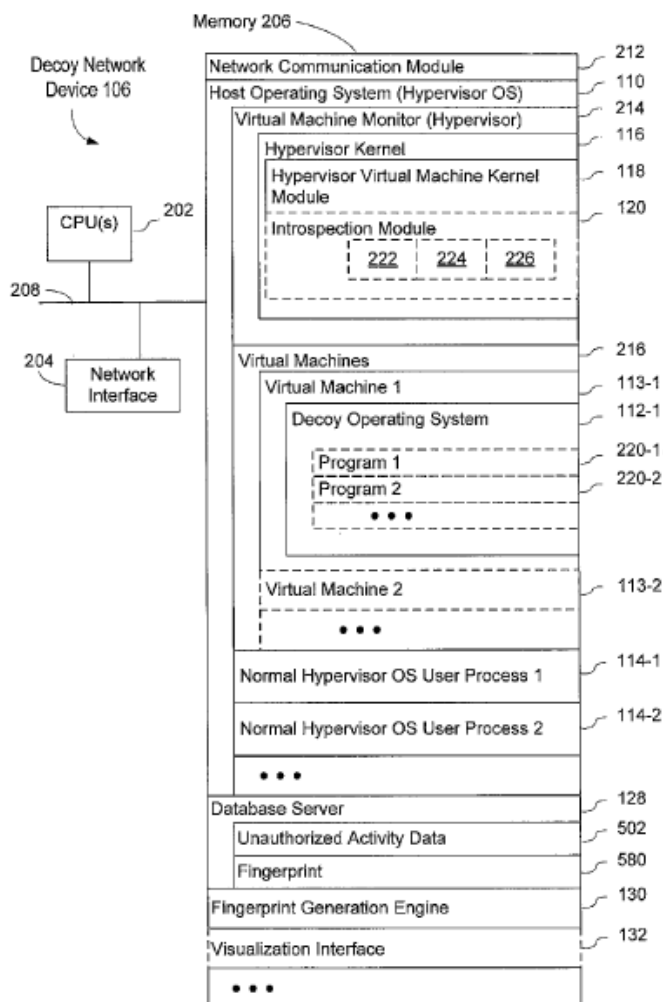


Figure 2

Id., Fig. 2. Decoy network device 106 includes one or more processing units (CPUs) 202, one or more network or other communications interfaces 204, memory 206, and communication buses 208 for interconnecting these components. *Id.*, [0033]. Memory 206 stores several programs, including, for example: a network communication module 212 used for connecting decoy network device 106 to computer networks and other computers; a host operating system 110 for handling basic system services and performing hardware dependent tasks; database server 128 for storing unauthorized activity data and/or fingerprints; fingerprint generation engine 130 for generating fingerprint data for the unauthorized activity data; and visualization interface 132 for displaying fingerprint data or unauthorized activities. *Id.*, [0035]-[0039].

72. In embodiments, the host operating system 110 includes a virtual machine monitor 214 (i.e., “hypervisor”); virtual machines 216; and normal hypervisor OS processes (e.g., administrative utilities, network tools, etc.). *Id.*, [0041]-[0044]. Virtual machine monitor 214 utilizes a hypervisor kernel 116 comprising a hypervisor virtual machine kernel module 118 to support virtualization of a “guest” decoy operating system. *Id.*, [0046]. Hypervisor kernel 116 uses an “introspection module 120” interposed between virtual machine monitor 214 and the decoy operating system 112 to monitor the physical memory segments assigned to decoy operating system 112. *Id.*, [0046]-[0048]. “In use, the introspection module

captures (through the introspection) raw attack information.” *Id.*, [0063]. “The raw attack information is then communicated to and stored on the database server 128 as forensic data 236 for later analysis.” *Id.*; *see also id.*, [0076]-[0089] (describing types of attack information stored, including activity sources, targets, actions, and associations (i.e., malignant or not); *see also id.*, Fig. 5A (depicting block diagram illustrating exemplary data structures of unauthorized activity data)).

73. While *Capalik* describes modules effectuating separate functions, *Capalik* discloses its’ modules “need not be implemented as separate software programs, procedures or modules, and thus various subsets of these modules may be combined or otherwise re-arranged in various embodiments.” *Id.*, [0054]; *see also id.*, [0055] (“One of ordinary skill in the art will recognize that an actual implementation might have the functional elements grouped or split among various components. In practice, and as recognized by those of ordinary skill in the art, items shown separately could be combined and some items could be separated”). *Capalik* further explains that “[a]lthough the methods described herein are described with respect to monitoring the decoy network device 106, persons having ordinary skill in the art would recognize that similar methods may be used for monitoring one or more non-decoy systems, such as the protected network device 136.” *Id.*, [0032]; *see also id.*, [0067] (“analogous methods can be used to represent and analyze unauthorized activities on non-virtualized computer systems”).

74. As discussed above, the '045 Patent is directed generally to methods for providing computer security by analyzing event data from potential threat events. *See above*, ¶¶24-27 (citing '045 Patent at 4:42-62, 4:63-5:19, 9:39-49, 23:4-13, claims 1, 8, 17, and 19). Similarly, as I described above, *Capalik* describes methods for analyzing malicious activity data on a computer system. *Capalik*, [0004] (“methods for analyzing malicious activities on a computer system”), [0030] (describing system and methods for collecting fingerprint data of unauthorized activities), [0048] (describing the generation of and retrieval of “forensic data (or fingerprint)”).

75. The '045 Patent is also directed at a problem of forensically analyzing the impact of an attack on a target device. *See*, '045 Patent, 1:15-18 (describing methods for “providing forensic visibility”), 1:22-24 (describing network forensics), 1:63-2:19 (describing malware analysis system with end-to-end visibility of system events). Likewise, *Capalik* provides for the “detailed forensic analysis of the character and nature of the attack and its impact on the target operating system.” *Capalik*, [0063]; *see also id.*, [0048] (describing analyzing “forensic data” from activities occurring on a device). *Capalik* is therefore reasonably pertinent to a problem faced by the inventor.

G. Opinions Regarding Ground 1: Claims 1-2, 4, 8-10, 12, and 16 are rendered obvious by Morris or alternatively, Claims 1-4, 8-10, 12, and 16 are rendered obvious by Morris in combination with Van Oorschot

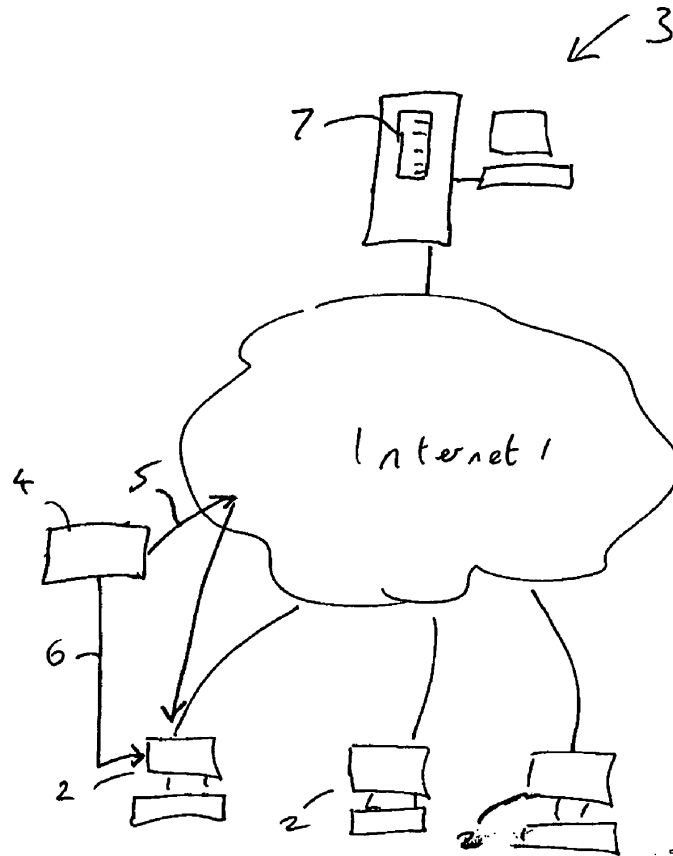
1. Claim 1

(a) *1[P]. A method comprising:*

76. *Morris* generally describes “methods and apparatus for dealing with malware” where such methods include “a “**method** and apparatus for classifying a computer object as malware.” *Morris*, [0001]. *Morris* explains that in embodiments, its method comprises the following steps: “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; comparing in the base computer the data about the computer object received from the plural computers; and, classifying the computer object as malware on the basis of said comparison.” *Id.*, [0014]-[0016]. *Morris* therefore discloses a “method.”

(b) *[1(a)] gathering one or more events defining an action of a first object acting on a target;*

77. *Morris*’s methods include a step for *gathering one or more events defining an action of a first object acting on a target*. For example, *Morris* describes that in its methods, “the data about the computer object that is sent from the plural remote computers to the base computer and that is used in the comparison includes one or more of...**events** initiated by or involving the object when the object is created, configured or runs on the respective remote computers.” *Morris*, [0018]. This method step of gathering event data is performed generally over the following architecture:



Id., Fig. 1.

78. Regarding the architecture of Figure 1 above, “base computer 3 holds a database 7 with which the remote computers 2 can interact when the remote computers 2 run an object 4 to determine whether the object 4 is safe or unsafe.” *Id.*, [0078]. “The community database 7 is populated, over time, with information relating to each object run on all of the connected remote computers 2.” *Id.* “[D]ata representative of each object 4 preferably takes the form of a so-called signature or key relating to the object and its effects.” *Id.*; *see also id.*, [0081] (the “details of the object are passed over the Internet 1 or other network to the base computer 3 for

storing in the community database 7 and preferably for further analysis at the base computer 3”).

79. *Morris* explains that the “details” associated with running the object on the plurality of remote computer 2 include “events.” *Id.*, [0087]; *see also id.*, [0104]-[0105] (describing capturing metadata associated with the object, including “Events”). With respect to these “events,” *Morris* explains that “these define the actions or behaviours of an object acting upon another object or some other entity.” *Id.*, [0105]. “The event has three principal components: the key of the object performing the act (the ‘**Actor**’), the act being performed (the ‘**Event Type**’), and the key of the object or identity of an other entity upon which the act is being performed (the ‘**Victim**’).” *Id.* These three components of an event are illustrated in the table below:

First Object	Action	Target
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. I have annotated the claimed first object (actor), action (event type), and target (victim), above, boxed in red.

80. I note that *Morris*’s described method step of gathering the three types of event data shown in the annotated table above mirrors and is consistent with the

described methods steps in the body of the '045 Patent. For instance, the '045 Patent describes its methods for gathering event data, including an *identical* table of event information. *See, e.g., '045 Patent* at 7:29-64:

TABLE 1

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

In fact, I note that the '045 Patent explains the system filters which collect and gather the event information for its disclosed methods of malware detection “may be built upon the same or similar technology related to behavior monitoring and collection, as discussed in U.S. application Ser. No. 13/372,375 filed Feb. 13, 2012, ‘Methods and Apparatus for Dealing with Malware’ (US-2012-0260340-A1)[.]” *'045 Patent*, 2:39-45. I refer to this publication in my report as the “*'340 Publication*” (Ex. 1009). The *'340 Publication* is authored by the same author as *Morris* relied on herein for my opinions—Melvyn Morris. The *'340 Publication* incorporates by reference *Morris*, the primary reference discussed in my opinions in this ground. *'340 Publication*, [0010] (“We refer in the following to our previous application US-A-2007/0016953, published 18 Jan. 2007, entitled ‘METHODS AND APPARATUS FOR DEALING WITH MALWARE,’ the entire contents of which are hereby

incorporated by reference”). The ‘340 *Publication* describes the general framework for gathering event data in its invention, such as in paragraph [0056]. ‘340 *Publication*, [0056] (depicting the same table of event data as found in both *Morris* and the ‘045 Patent). Indeed, up until [0064], the ‘340 *Publication* admits the “system as described so far is generally described in our previous application US-A-2007/0016953,” which of course refers to *Morris*.

81. It is my opinion that *Morris* discloses *gathering one or more events defining an action of a first object acting on a target* using the same techniques as the ‘045 Patent.

(c) [1(b)] *generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object,*

82. On top of the event metadata associated with running an object at one or more remote computers, *Morris* describes additional types of context generated from an event that is sent to the base computer to further inform analytical processing techniques. For example, *Morris* discloses that “[a]mongst other types, the types of metadata captured might be” (*see id.*, [0104]): “‘**Genesisactor**’—the key of an object that is not the direct Actor of an event but which is **the ultimate parent of the event being performed.**” *Id.*, [0107]. For example, “in the case of a software installation, this would be the key of the object that the user or system first executed and that initiated the software installation process, e.g. Setup.exe.” *Id.*

83. Just like *Morris's* disclosure of gathering event data including the claimed first object (actor), action (event type), and target (victim), described above, I note that *Morris's* process for gathering “Genesisactor” data (i.e., data concerning the ultimate parent of the event being performed) is likewise consistent with the '045 Patent's described process for *generating a contextual state...by correlating the at least one event to an originating object*. In particular, the '045 Patent describes its process for generating a contextual state as follows:

“contextual state information may include GenesisActor information...GenesisActor information may contain the object that is not the direct Actor of an event but which is the ultimate parent of the event being performed. For example in the case of a software installation, the GenesisActor would be the object that the user or system first executed and that initiated the software installation process, e.g. Setup.exe.”

'045 Patent, 8:42-51. Further, the '045 Patent explains that to generate a contextual state, a contextual analysis is applied to the captured events such that additional data is gathered supplementing or enhancing the event data, including Genesisactor data. *See, e.g., '045 Patent, 17:30-35* (“a contextual analysis may be applied to the captured events 504B-520B. Accordingly, a context 520B comprising an origination process [sic], Device, User, and in some instances, a GenesisActor may further supplement or enhance the Event ID, Event Type, Actor, and Victim”).

84. The '045 Patent describes this process of correlating an object to an originating process, device, or user as forming an “audit trail” for an event. '045

Patent, 12:55-63 (“Accordingly, the context analyzer 348 correlates objects within system events to additional objects known to exist. For example, the context analyzer 348 may correlate an actor, victim, and/or event type to one or more originating processes, devices, and users. Such a correlation creates an audit trail for each individual event.”). *Morris*’s disclosure above concerning ancestral processes, such as the Genesisactor for an event, is likewise consistent with the ’045 Patent’s “audit trail” discussion. *See also Morris*, [0052] (“This aspect uses the concept of ancestry to enable objects to be marked as malware. For example, any particular process may spawn child processes which are therefore related...”), [0099] (“For example, analysis can be made of which object created this object, which objects this object created...In this regard an object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object. This simple 1-to-1 relationship chain...”).

85. It is my opinion, therefore, that *Morris* discloses *generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object* using the same techniques as the ’045 Patent.

- (d) *[1(b)(i)] the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object;*

86. I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to

render this limitation unpatentable, where this limitation claims “at least one of” a “device on which the first object is executed and a user associated with the first object[.]” In other words, to render this limitation unpatentable, the prior art need only disclose one of the two: the device where the first object is executed or a user associated with the first object.

87. As I described in my opinions in the preceding limitation, *Morris* describes additional types of context generated from an event, including the Genesisactor—i.e., the ultimate parent of the event being performed. Consistent with my opinions above, the “Genesisactor” discloses the *contextual state including an indication of the originating object of the first object*. More specifically, as I described above, *Morris* discloses generating Genesisactor information in a software installation process. *See, e.g., Morris*, [0107] (“in the case of a software installation, this would be the key of the object that the user or system first executed and that initiated the software installation process, e.g. Setup.exe”). In this example, I understand the Genesisactor actor is the Setup.exe executable file. A person having ordinary skill in the art would have understood that the Setup.exe executable file serves as the *originating object of the first object*. Specifically, Setup.exe is a common Windows Installer launchpad used to run an executable program. Typically, this program is MSIEXEC.EXE. MSIEXEC.EXE processes the setup package in order to facilitate installation of a software product. Here, MSIEXEC.EXE is the

first object which performs the action of installing software. Accordingly, the Genesisactor—Setup.exe—is the *originating object* which runs the MSIEXEC.EXE *first object* to effectuate software installation.

88. Similarly, a skilled artisan would have understood in any of the other disclosed events in *Morris*, generation of a Genesisactor would also serve as the *originating object of the first object*. For example, the captured events are re-illustrated below in *Morris*'s table:

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., [0105]. I have highlighted the second event for simplicity. In this second event, Object 1 (first object) sends data to a target at IP Address 3. *Morris*'s step of generating Genesisactor information would include generating the ultimate parent of this event. I note this would include, for example, the executable file which created Object 1. And as such, the Genesisactor here likewise indicates the *originating object of the first object*.

89. I additionally note that in the example second event above, *Morris*'s generation of a contextual state additionally includes *an indication of...a device on*

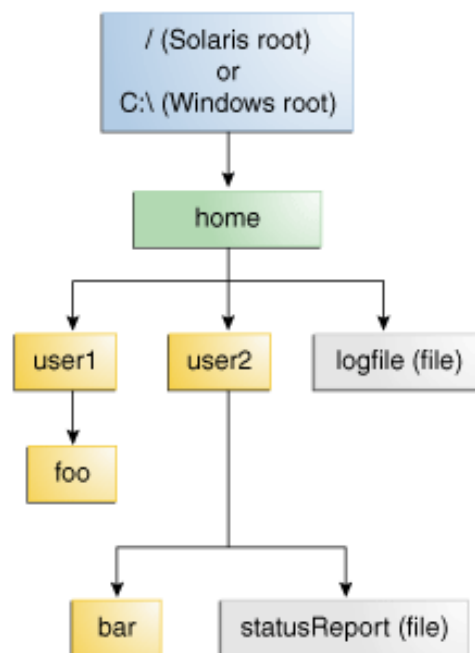
which the first object is executed. For example, the device where the first object is executed in this second event is illustrated below:

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

Morris, [0105]. I have boxed in red the device where the first object is executed (i.e., the device at IP Address 3). In *Morris*, the “Victim” is referred to as the “identity of an other entity upon which the act is being performed[.]” *Id.* In this example, the Victim is the device at IP Address 3 which in turn represents the device on which data is sent—i.e., the first object, “Object 1,” is executed on the device identified as “IP Address 3” such that data is sent to this device. A skilled artisan would have understood this IP Address *indicates* the device because an IP address identifies a device on the global internet, acting as the device’s logical address to identify that network connection. The IP Address is a 32-bit number uniquely identifying this device, normally displayed in dotted decimal notation, e.g., 123.456.7.8.

90. Additionally, *Morris* also describes that other contextual information can be generated from an event, including “Identities” which serve to “define the attributes of an object,” such as “its logical location on the disk within the file system (its path).” *Morris*, [0106]. A skilled artisan would have understood that when

generating information concerning a file's "path," *Morris's* disclosed methods would include generating *an indication of at least one of a device on which the first object is executed or a user associated with the first object*. A "path" is the address of an object (e.g., file) on a filesystem. This filesystem is a hierarchy of directories used to organize files on a computer. The following shows a sample directory tree for Microsoft Windows and Solaris OS:



Sample Directory Structure

The Java Tutorials, What is a Path? (And Other File System Facts) (Ex. 1021). A file, such as the "statusReport" file, is identified by its path through the file system, beginning from the root node. For example, this file's path in Microsoft Windows would read as follows:

C:\\home\\sally\\statusReport.

“Sally” as emphasized in red in the above filepath indicates the user associated with the file.

91. There are two basic types of paths, relative paths and absolute paths. A relative path is the location of a file relative to the current directory where the user is currently working. An absolute path (also called a “full path”) is the location of a filesystem object relative to the root directory. The absolute path *always* contains the root element and the complete directory list required to locate the file. The above filepath is an “absolute path” because all of the information needed to locate the file is contained in the path string.

92. A filepath can also include the device where the file is executed. For example, a file downloaded on a user’s desktop could include the following filepath:

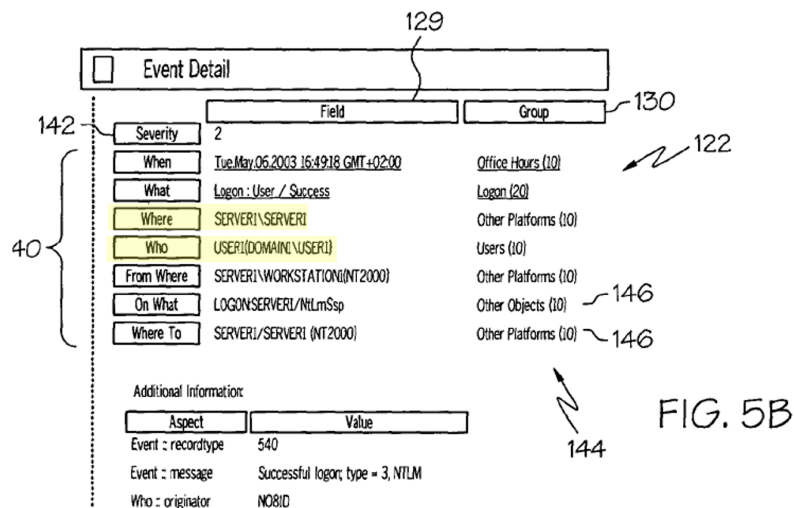
C:\Users\Sally\Desktop\filename

Here, the device is indicated as Sally’s desktop computer.

93. Should Patent Owner or its’ expert suggest *Morris* does not disclose *an indication of at least one of a device on which the first object is executed and a user associated with the first object*, it is my opinion that *Van Oorschot* does. As I described above in my overview of *Van Oorschot*, *Van Oorschot* generally relates to a similar computer security system involving methods for gathering and analyzing event data. In this method, a “computer receives a system event initiated by an initiating client associated with a user[,]” “[t]he system event comprises a plurality

of data elements associated with respective ones of a plurality of system event attributes[,]” and then this system event is compared against a security policy to determine a severity level for the event. *Van Oorschot*, 1:55-2:15, 3:29-31.

94. To accomplish this method, *Van Oorschot* discloses an “event management tool” operable to “record, manage, and respond to, system events that occur on the local network.” *Id.*, 3:29-31. The event management tool normalizes the event data gathered by the system, including “‘who’ (the user initiating the event)” and “‘where’ (the system or device on which the event occurred)[.]” *Id.*, 6:14-23. For example, this normalized event data is illustrated below in Figure 5B



Id., Fig. 5B. I have highlighted the “where” and “who” event records which in my opinion indicate the device on which the first object is executed (“Where”) and the user associated with the first object (“Who”). I have also highlighted this same event information in Figure 5A below, which represents the event data prior to normalization by the event management tool:

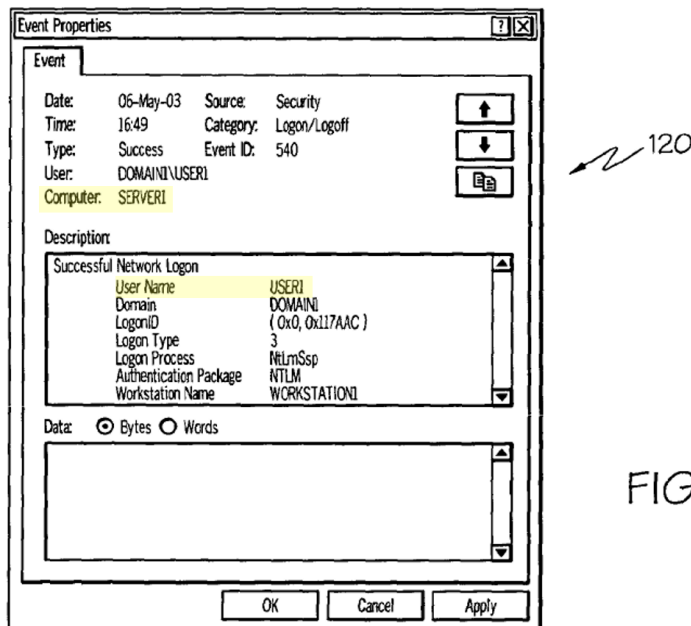


FIG. 5A

Id. Fig. 5A.

95. The above event data comes from an example system event of a Windows/NT 2000™ fileserver. *Id.*, 4-13. In this event, USER1 logs into a Windows/NT 2000™ fileserver from his or her own workstation, WORKSTATION1. The event occurs on a machine SERVER1. *Id.* Thus, in the normalized event log, USER1 is populated in the “Who” record which relates to the user who logged into the fileserver and SERVER1 is populated in the “Where” record relating to the device where the event occurred at. *See also, Van Oorschot*, 6:44-7:6 (describing another example pertaining to a system event of a Checkpoint Firewall-1 Firewall™ server), Figs. 5B and 6B.

96. In my opinion, it would have been obvious to a person having ordinary skill in the art to configure *Morris’s* system filters to capture additional types of

event metadata, such as the event data disclosed in *Van Oorschot* concerning “where” (i.e., “the device where the event occurred”) and “who” (i.e., “the user initiating the event”). For example, those of skill in the art were well-aware that gathering more detailed information from threats results in improved threat analysis and detection. In particular, analogous art in the field, such as *Polyakov*, expressly indicates that “keeping additional information about threats than what was previously available” provides “a more complete threat model to the back-end.” *Polyakov*, [0028]. “[L]imited metadata” may be “insufficient for back-end analysis” and thus “[t]o improve the quality and success ratio of back-end threat analysis,” malware analysis systems need to “collect[] more meta-data from the infected computer.” *Id.*, [0029]; *see also above*, ¶¶ 41-44 (citing *Evolving Endpoint Malware Detection: Dealing with Advanced and Targeted Attacks* by Securosis (Ex. 1014) for this same principle, which further notes that “efforts to detect malware can no longer focus exclusively on what malware looks like (a file hash or some other identifying factor) and must incorporate a number of new data sources for identification”).

97. Consistent with this reasoning, a skilled artisan would have been motivated to collect device and user information associated with an event in *Morris*’s system to predictably improve the quality and success of *Morris*’s back-end threat identification and analysis. A skilled artisan would have also been motivated by

Morris's express teachings. *Morris* contemplates deriving additional metadata “not specifically mentioned” in its disclosure. *Morris*, [0110]. This includes any combination of disclosed metadata elements and additional elements not mentioned therein. *Id.*; *see also id.*, [0104] (“[a]mongst other types” phrase also indicating other metadata not expressly described in *Morris* may be collected).

98. Because it was both a common principle in the art to collect additional metadata and *Morris* expressly contemplates collecting metadata not specifically mentioned in its disclosure, there would have been a reasonable expectation of success modifying *Morris* to collect such additional metadata. This combination of prior art elements would have been accomplished using known methods and with no change to their respective functions. For example, in the proposed modification, *Morris*'s system would derive the disclosed metadata associated with a threat event, such as the actor, event type, victim, Genesisactor, and would simply derive additional device and user metadata associated with this event, as described in *Van Oorschot*. As I discussed above, generating additional information about threats was a common way to provide more complete threat models for back-end analysis, and thus there would have been a reasonable expectation of success configuring *Morris*'s system to collect device and user information along with the other types of metadata from a threat event to better inform the backend analysis.

- (e) [1(c)] obtaining a global perspective for the at least one event by obtaining information associated with one or

more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information,

99. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “the information including at least one of” an “age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information[.]” To render this limitation unpatentable, the prior art need only disclose one of the items in the above list.

the information including at least one of...Internet Protocol (IP) Address

100. As discussed in my opinions above in the preceding limitations, in example event two highlighted in *Morris’s* table at [0105], *Morris* discloses obtaining the IP Address of the device that receives data from the first object. *See above*, ¶¶86-92 (noting this is the device where the first object is executed). This table is reproduced below:

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

Morris, [0105]. In my opinion, obtaining this IP Address discloses the claimed *obtaining a global perspective for the at least one event by obtaining information associated with...the first object...the information including at least...Internet Protocol (IP Address)*. The first object (“Object 1”) sends data to the device identified as “**IP Address 3**” boxed in red. Obtaining the IP Address of the device Object 1, i.e., the *first object*, sends data to therefore represents information associated with the first object.

the information including at least one of... a determination as to whether the first object is malware, a determination as to whether the originating object is malware

101. *Morris*’s methods generally include the following: “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; comparing in the base computer the data about the computer object received from the plural computers; and, **classifying the computer object as malware on the basis of said comparison.**” *Id.*, [0014]-[0016]. Therefore, after event data is gathered and contextual data is generated as I described in the preceding limitations, *Morris* sends this information to a base computer for further analysis. *Id.* This further analysis “allows a comparison to be

made between the objects and/or their effects on the different remote computers **to determine whether or not a particular object should be classed as good or as malware.**” *Id.*, [0017]; *see also id.*, [0081]-[0084] (explaining process for making a determination that an object is “bad or unsafe”). This determination thus includes a *determination as to whether the first object is malware.*

102. *Morris* further discloses this comparison between objects can include identifying “relationships between the object and other objects” such that the system can “immediately and automatically to mark a child object as bad (or good) if the or **a parent or other related object is bad (or good).**” *Id.*, [0019]. “[I]f at least one other object to which said object is related is classed as malware, then the method may comprise classifying said object as malware.” *Id.* “Said other objects include the object or similar objects stored on at least some of the remote computers.” *Id.* **“Said other objects may include other objects that are parent objects** or child objects or otherwise process-related objects to said object.” *Id.* *Morris* therefore also discloses a *determination as to whether the originating object is malware.*

103. This malware determination for related objects is described in more detail in *Morris* as follows. *Morris* describes a “method of classifying a computer object as malware or as safe, wherein said computer object is a descendant or otherwise related object of a first computer object, the method comprising:”

classifying a first computer object as malware or as safe; identifying in a key relating to said first computer object a component that

uniquely identifies the first computer object and that is inherited or otherwise present in the key of a descendant or other related computer object of the first computer object; and, classifying said computer object as malware or as safe as the case may be on the basis of the unique identifier component being present in the key of said computer object.

Id., [0048]-[0051]. *Morris* explains this “aspect uses the concept of ancestry to enable objects to be marked as malware.” *Id.*, [0052]. “For example, any particular process may spawn child processes which are therefore related.” *Id.* “The key relating to the first object may be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object.” *Id.*

104. *Morris* also discloses five “heuristics”/“processes” used alone or in combination to effectuate “the rapid identification and **determination of malware.**” *See, e.g., id.*, [0096]-[0103]. The second of these processes uses “an object’s relationship to other objects that act upon it or upon which it acts” in order to make a malware determination. *Id.*, [0099]. An “analysis can be made of which object created this object, which objects this object created...” so that “[o]bjects that have specific relationships, volumes of relationships or mixes of relationships to one type or another may be judged safe or unsafe accordingly.” *Id.* “[A]n object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object.” *Id.* “This simple 1-to-1 relationship chain provides a complex series of correlation points, allowing ancestral relationships to be

considered for any object and its predecessors by event or its issue (i.e. child and sub-child processes) by event.” *Id.*

105. *Morris* provides an example of using object relationships to make a malware determination. Consider for example “notepad.exe as supplied by Microsoft with the Windows series of operating systems.” *Id.*, [0099]. This file has a limited range of functionality and would not be expected to perform a wide variety of events, such as transmitting data to another computer or running other programs etc. *Id.* However, the behavior of notepad.exe could be modified by injecting new code into it, such as via dynamic link library injection (DLL injection). *Id.* In this case notepad.exe would now have new capabilities derived by the code injection or linkage to another object. *Id.* Using the data that defines the relationships between objects, it is possible to deduce that the new behaviors of a program can be attributed to the association with another object. *Id.* If that new behavior is malevolent, then it is possible to mark either or all processes as unsafe as appropriate. *Id.*

106. This “ancestry” or “gene” heuristic provides for both the child object and its parent or any other object in the lineage to be determined as malware. *Id.*, [0128]-[0134]. By “monitoring activity of the parent process, if it is later found that the parent process is malware, then all of the issue processes can all automatically be recorded in the local database as bad.” *Id.*, [0130]. Analogously, by monitoring activity of the issue processes, “if it is later found that one of the issue processes is

malware, then one or more of the parent process and all of the other issue processes (i.e. all of the related processes in this context) can all automatically be recorded in the local database as bad.” *Id.*, [0131].

the information including at least one of... popularity

107. In addition to the second heuristic above relating to a malware determination for a first and/or originating object, *Morris* discloses a third heuristic involving “time and **volumes**.” *Morris*, [0101]. In this heuristic, “[r]elevant data includes when the object was first seen, when it was last seen, **how many times it has been seen, how many times it has been seen in a given interval of time**, and the increase or decrease of acceleration in it being seen.” *Id.* “This information is highly relevant in determining the prevalence of an object in the community of remote computers.” *Id.* These metrics are further used to “determine if an object is safe, unsafe or too prevalent to allow it to execute or propagate without very thorough examination.” *Id.*

108. In my opinion, *Morris* therefore additionally discloses obtaining a global perspective for the event with information concerning the *popularity* of the first object.

109. *Morris* determines this an object’s popularity in the same way as the ’045 Patent. For example, the ’045 Patent describes a process where “[f]or unknown files, the dashboard may show their popularity at the global community, i.e. on how

many computing systems 204 they have been seen.” ’045 Patent, 16:15-18.

Similarly, as I described above, *Morris*’s third heuristic takes into account “how many times it has been seen, how many times it has been seen in a given interval of time.” *Morris*, [0101].

(f) *[I(c)(i)] wherein the global perspective for one or more related events to the at least one event across a network;*

110. For the purposes of my analysis, I have been asked to interpret this limitation to require the global perspective include “information about events across the network” where those events occur on “other devices” on the network.

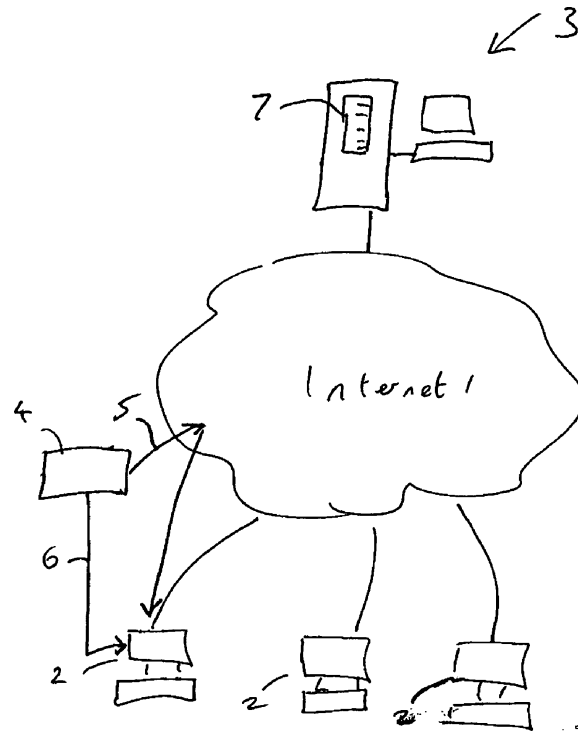
111. *Morris*’s methods for obtaining a global perspective involve a comparison of related events provided by other computers. Specifically, *Morris* describes running an object on a plurality of remote computers and sending that data from the plurality of remote computers to a base computer for comparison and analysis. *Morris*, [0014]-[0016]:

According to a first aspect of the present invention, there is provided a method of classifying a computer object as malware, the method comprising: 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; **comparing in the base computer the data about the computer object received from the plural computers; and, classifying the computer object as malware on the basis of said comparison.**

See also id., [0067] (method “comprises connecting the computer to a community database that is connectable to a plurality of computers, and **uploading the stored**

data to the community database for comparison with similar data provided by other computers”).

112. In particular, this process is shown below in Figure 1:



Id., Fig. 1. Each of the remote computer 2 run object 4 and send the details from this over the Internet 1 to a base computer 3 comprising community database 7. This information is stored in community database 7 and base computer 3 can analyze this information as described above. *See above*, ¶¶101-106 (describing heuristics to inform a malware determination and popularity determination). The “comparison with similar data provided by other computers” is therefore conducted at base computer 3. And this comparison includes “classifying the computer object as malware on the basis of said comparison” (*see id.*, [0016]) and making a popularity

determination based on the “prevalence of an object in the community of remote computers” (*see id.*, [0101]).

113. Additionally, as I explained above, obtaining a global perspective also includes determining an IP address associated with a first object. One of skill in the art would have understood this IP address would similarly include information about events across a network. In the example above, Object 1 sends data to a device at IP Address 3. In the community-based analysis, this same Object 1 would be run on each of the plurality of remote computers such that a determination of the IP address of the device data is sent to would be obtained in each instance. In this way, determining the IP address from one event would relate to the similar determination of an IP address from another event. Accordingly, the global perspective in this scenario (i.e., obtaining the IP address of the device data is sent to) would relate to one or more related events across a network.

- (g) *[1(d)] assembling an event line including details associated with the at least one event, the details including information uniquely identifying the first object, the action of the first object, the target, and the originating object; and*

114. *Morris* describes that its base computer 3 holds a database 7 which is “populated, over time, with information relating to each object run on all of the connected remote computers 2.” *Morris*, [0078] (remote computers 2 “run an object 4” and gather and transmit “information relating to each object run on all of the

connected remote computers” to community database 7); *see also id.*, [0018]. This data includes the event and contextual data described in my opinions above. The “data representative of each object 4 preferably takes the form of a so-called signature or key relating to the object and its effects.” *Id.*, [0078]. “This key may be formed by a hashing function operating on the object at the remote computer 2.” *Id.*, [0086].

115. *Morris’s* key includes the *details associated with at least one event, including information uniquely identifying the first object, the action of the first object, the target, and the originating object.* Specifically, *Morris* discloses that its’ key includes “data about said object” such as “events initiated by or involving he object when the object is created, configured or runs on the respective remote computers” *Id.*, [0087]. And as I explained above, this event data includes “the key of the object performing the act (the ‘Actor’), the act being performed (the ‘Event Type’), and the key of the object or identity of an other entity upon which the act is being performed (the ‘Victim’).” *Id.*, [0105]. *Morris* further discloses that additional event data is captured and sent to community database for analysis, including the “‘Genesisactor.’” *Morris*, [0107], [0110] (describing that all of the derived metadata is used at the community database 7 to define the behavior of a process across the community). A skilled artisan would have understood this Genesisactor information would be sent in the key along with the other event information (e.g., the actor, event

type, and victim, for transmission to the community database for subsequent analysis. *See, e.g., Morris*, [0086] (noting the “details of an object 4 that are passed to the base computer 3 are preferably in the form of a signature or ‘key’” in order to “keep down the data storage and transmission requirements”).

116. In my opinion, *Morris* therefore discloses *assembling an event line including details associated with the at least one event (key) the details including information uniquely identifying the first object (Actor) the action of the first object (Event Type) the target (Victim) and the originating object (Genesisactor).*

(h) *[1(e)] transmitting the assembled event line.*

117. As I described above in ¶¶114-116, *Morris*’s remote computers run an object and gather and transmit information relating to each object run on all of the connected remote computers to the base computer comprising a community database. *See also Morris*, [0020] (describing sending data in “the form of a key” to keep “down the volume of data that needs to be transmitted to the base computer”), [0123] (“The key is transmitted to the community database 7), Fig. 1 (depicting network data is transmitted over). Furthermore, this key can be subsequently “copied to” the local database of a remote computer that first ran the object. *Morris*, [0084].

2. *Claim 2 – The method of claim 1, wherein the information associated with the one or more of the first object and the originating object is obtained from a community database; and wherein the event line is transmitted to an end recipient.*

118. As explained in my opinions above, *Morris* discloses obtaining information associated with the first object and/or originating object in four ways: (1) obtaining an IP Address of the device where Object 1 is executed; (2) obtaining a determination that the first object is malware; (3) obtaining a determination that the originating object is malware; and (4) obtaining a popularity. As I also explained above, each of these types of information can be obtained from a community database. For instance, *Morris* discloses five heuristics which are “run on the data in the **community database 7**.” *Id.*, [0095], [0110].

119. With respect to (1) obtaining an IP address, the fourth heuristic “considers the behaviour of an object” from data stored in community database 7, including the “victim” which includes identification of an IP address. *Id.*, [0102], [0105].

120. With respect to (2) and (3) the malware determinations for the first object and originating object, the second heuristic analyzes object relationships stored in community database 7 to make a malware determination. *Id.*, [0100] (“centralisation of the community data in the community database 7 provides the ability to rapidly assimilate object behaviours, allowing for the rapid identification and determination of malware”).

121. With respect to (4) popularity, the third heuristic analyzes “time and volume[]” data from community database 7 to determine “how many times [an object] has been seen[.]” *Id.*, [0101].

122. In my opinion, *Morris* therefore discloses the *information associated with the one or more of the first object and the originating object is obtained from a community database.*

123. Furthermore, *Morris* describes that its *event line is transmitted to an end recipient*. Specifically, *Morris* explains that its “key” (*event line*) is transmitted to base computer 3 (*end recipient*) for processing. *See above*, ¶117. As explained above in my Background of the Technology, it was well-known in the art to send signature-based data to backend central computer comprising a community-database for analysis. *See above*, ¶¶43-52. In particular, each of *Ye*, *Dozortsev*, *Chess*, *Shyne-Song Chuang*, *Spiegeleer*, and *Hinchliffe* and several research papers each describe community-based systems where data is sent to a backend central computer or server for analysis.

124. In addition to or alternatively, *Morris* describes that after the analysis at base computer 3 is performed, the key can be “marked as bad or unsafe [in the] community database 7” and then the “signature is copied to the local database of the remote computer 2 that first ran the object[.]” *Id.*, [0084]. To copy this data back to the local database at remote computer 2, one of skill in the art would have understood

this key would be “transmitted” to remote computer 2 over the network (e.g., Internet

1). Here, I understand that “remote computer 2” would serve as the *end recipient*.

3. *Claim 3 – The method of claim 1, further comprising: obtaining an event priority based on one or more of the generated context and the global perspective for the at least one event, wherein the event line further includes the event priority for the at least one event.*

125. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “obtaining an event priority based on one or more of” the “generated context and the global perspective for the at least one event[.]” To render this limitation unpatentable, the prior art need only disclose one of these items.

126. As explained in my opinions above in claim 1, *Morris’s* methods include generating contextual information and a global perspective for an event and generating a key (i.e., *event line*). *See above*, Claim 1. However, *Morris* does not explain that its methods obtain an event priority based on one or more of this contextual information or global perspective.

127. *Van Oorschot* describes a process for assigning a severity level to a system event, which I understand to refer to obtaining an event priority for events based on the assigned severity level. For example, *Van Oorschot’s* severity assignment generally operates as follows:

A computer receives a system event initiated by an initiating client associated with a user. The system event comprises a plurality of data elements associated with respective ones of a plurality of system event attributes. It is determined that the system event fails to conform to the security policy. A greatest of a plurality of significance factors assigned to a plurality of data groups that are associated with the plurality of data elements of the system event is determined. If the system event does not match a special attention rule, the greatest of the plurality of significance factors is assigned to the system event as a severity level. If the system event matches the special attentions rule, then a greater of the greatest of the plurality of significance factors and the pre-assigned severity level is assigned to the system event as a severity level.

Van Oorschot, Abstract; see also id., 1:55-2:15.

128. *Van Oorschot's* severity level is assigned *based on the generated context* related to the event. For example, Figure 11 below is a flow diagram for assigning a severity level to an event:

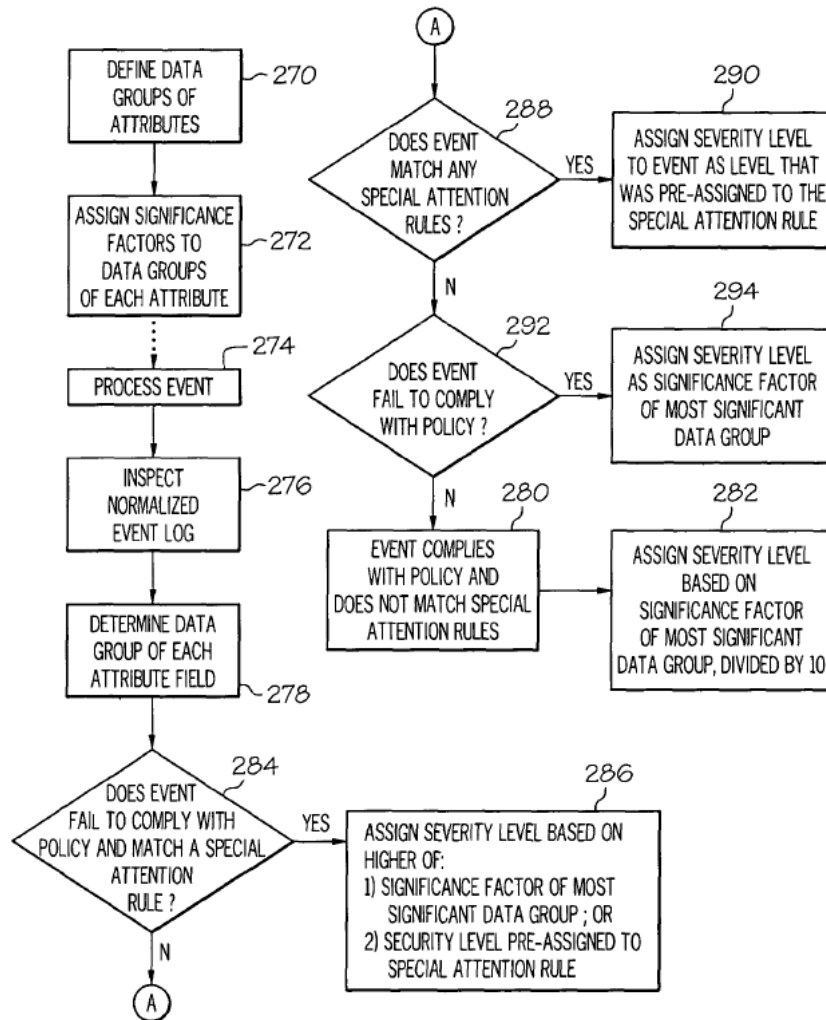


FIG. 11

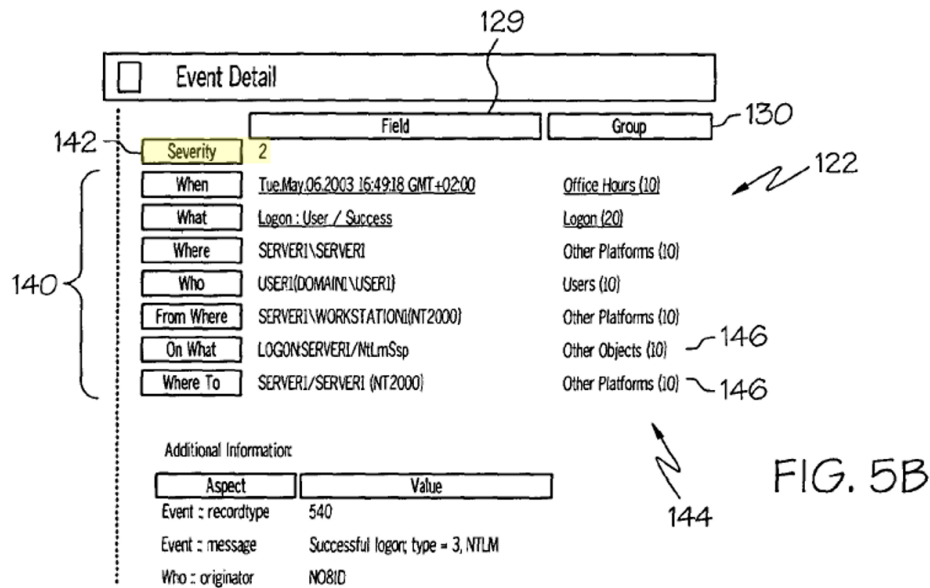
Id., Fig. 11. In this process, data groups of attributes are pre-defined, such as:

“when” attribute may include groups that define time elements, such as “office hours”, “weekends”, or “weekdays”. The “who” attribute may include groups related to the various forms of users, such as “network administrators” or “general users”. The “what” attribute may include groups related to the types of actions that can occur, such as “file write actions”, “file read actions”, “administrative actions”, or “email transfer”. The “on what” attribute may include, for example, groups related to the types of files being operated on. The “where”, “where from”, and “where to: attributes may include groups related to the types of clients at which the event can occur, the types of clients initiating the event, or the types of clients being

acted upon by the event, for example “web servers”, “test servers”, “mail servers”, “file servers”, “workstations”, or “database servers”.

Id., 7:7-27.

129. Then, “each group within an attribute is [pre]assigned a significance factor” that is used “in later assigning a severity level to an event[.]” *Id.*, 7:27-32. After the data groups are pre-assigned and significance factors are pre-assigned to the defined data groups, an event is processed, inspected, and the data groups of each attribute field are determined. *Id.*, 8:12-26. “Next, a severity level is assigned to the event according to whether the event complies with systems policies, and according to whether the criteria of any special attention rules are met.” *Id.* As shown in Figure 11 above, a severity level is assigned to an event based on these significance factors and rules—i.e., *based on the generated context* related to the event. *Van Oorschot’s* “severity level” for an event is illustrated below:



Van Oorschot, Fig. 5B. I have highlighted the severity level 142 assigned to this system event.

130. Thus, in sum, based on this process, “the system events are each **assigned a severity level**, based on exceptions or breaches identified during the comparison” and the system then “reacts to the event or events based on the severity level of the breach.” *Id.*, 5:39-44. I note that this process of obtaining an event priority by assigning a severity level is consistent with the ’045 Patent’s disclosure. *See, e.g., ’045 Patent*, 9:28-49 (describing “rule set” which “may determine that an event meeting certain conditions be **assigned a priority according to a scale of severity**”), 13:34-46, 22:27-52, Fig. 11.

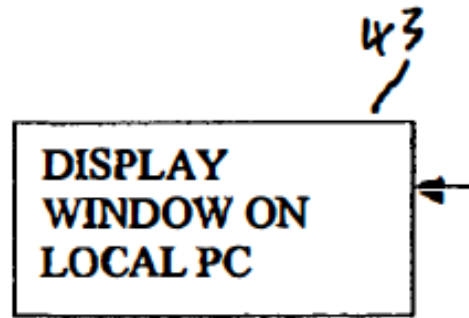
131. One of skill in the art would have been motivated to modify *Morris*’s malware detection methods to incorporate *Van Oorschot*’s known assignment of

severity levels to systems events to appreciate several benefits, including, for example, focusing threat analysis on the most troublesome threats and promoting the effective use of limited resources. Indeed, these benefits were well-appreciated in the art during the relevant time period. *See, e.g., Polyakov* at [0021] (prioritization of threats “improves threat detection on an on-going basis by focusing threat analysis resources on the most troublesome threats) and [0040] (priority functionality “promotes more timely inspection by technicians or automated analysis, thereby more effectively utilizing limited analysis resources”). Further, Van Oorschot describes express benefits, including explaining that this severity level “is indicative of the general significance of a given event” which can provide “an additional gauge by which a decision can be made as to the manner in which an event is to be processed or as to the relative importance of the event.” *Van Oorschot*, 8:4-11. Indeed, this severity level “can be used as a sorting parameter” for solutions. *Id.*, 9:66-10:1. Therefore, a skilled artisan would have understood that applying this known concept to *Morris’s* malware analysis system would be conducted according to well-understood methods to yield predictable results.

132. A skilled artisan would appreciate a reasonable expectation of success making this modification. Specifically, in *Morris’s* modified methods, once event information has been gathered, contextual information such as originating object and the device and/or user associated with the first object has been generated, and a

global perspective (e.g., IP Address associated with the first object, a malware determination for the first/originating objects, or a popularity) has been obtained, *Morris's* base computer would incorporate *Van Oorschot's* severity level assignment process to appreciate the benefits described above. One of skill in the art would have understood assigning "severity levels" based on, for example, the event and contextual data generated during processing an event includes *obtaining an event priority based on...the generated context*. Thus, when this threat information is transmitted back to a remote computer 2 which originally ran the object (*see Morris*, [0084]), the user running the object would be able to focus subsequent malware remediation attempts on the most important threats that need prompt attention (i.e., threats with high severity levels) while conserving limited resources.

133. It would have been obvious to a skilled artisan to include the event severity level (i.e., *priority*) in *Morris's* key (i.e., *event line*) to highlight to a user the threats which need prompt attention as discussed above. *Morris's* local computers come equipped with a "display window" for presenting information to a user, *see Morris*, [0126], illustrated below:



Id., Fig. 4 (excerpt). Therefore, it would have been simple to modify *Morris* to display this information on this display window. This modification would be conducted according to well-known methods. For example, *Van Oorschot* expressly contemplates displaying these event severity levels to a user to inform threat remediation tactics. *See, e.g., Van Oorschot*, Figs. 7-10.

134. Additionally, U.S. Patent Publication No. 2004/0143753 to Hernacki et al. ("*Hernacki*") (Ex. 1022) discloses a system and method for analyzing security risks in a computer network, where the "system constructs asset relationships among a plurality of objects in the computer network and receives an event associated with a selected object, where the event has an event risk level." *Hernacki*, Abstract, Figs. 4-5 (describing method of assigning event risk level). This "risk level" is "calculated based on a variety of factors including the attribute information of the object (including object type, location, trust relationships, vulnerability state, etc.), events that have been detected at the object, events from other objects on which this object

depends, and time dependent functions, etc.” *Id.*, [0021]; *see also id.*, [0035], [0038] (discussing transmitting this risk level with the event).

135. Moreover, U.S. Patent Publication No. 2011/0083180 to Mashevsky et al. (“*Mashevsky*”) (Ex. 1023) describes a system and method for detection of previously unknown malware, comprising:

(a) receiving event information and file metadata from a remote computer; (b) identifying whether the event information or the file metadata are indicative of the already known malware presence, indicative of the unknown malware presence, or indicative of malware absence; (c) if the event information or the file metadata are indicative of the known malware or indicative of malware absence, filtering out the event information and the file metadata; (d) performing a risk analysis and risk assessment for the remaining event information and the remaining file metadata to determine if the event and the file metadata are indicative of the previously unknown malware presence; and (e) where performing a risk analysis and risk assessment includes a "parent-child" hierarchy of the files, and the risk assessed to the parent is based on the risk associated with the child.

Mashevsky, Abstract. As part of this method, *Mashevsky* explains that its system calculates an “X-factor” which “defines the level of danger of a given ‘parent’ and is based on the data about its ‘children’” (*see id.*, [0047]) and a “Significance” value which is “a function that takes into account the level of activity and the degree of danger of an event” (*see id.*, [0049]).

136. Furthermore, *Polyakov* is directed to a similar “malware analysis system” that “provides information about malware execution history on a client computer and allows automated back-end analysis for faster creation of

identification signatures and removal instructions.” *Polyakov*, [0006]. This malware analysis system operates as follows:

The malware analysis system collects threat information on client computers and sends the threat information to a back-end analysis component for automated analysis. The back-end analysis component analyzes the threat information by comparing the threat information to information about known threats. The system builds a signature for identifying the threat family and a mitigation script for neutralizing the threat. The system sends the signature and mitigation data to client computers, which use the information to mitigate the threat.”

Id.

137. Figure 2 below is a block diagram illustrating a typical operating environment of the malware analysis system:

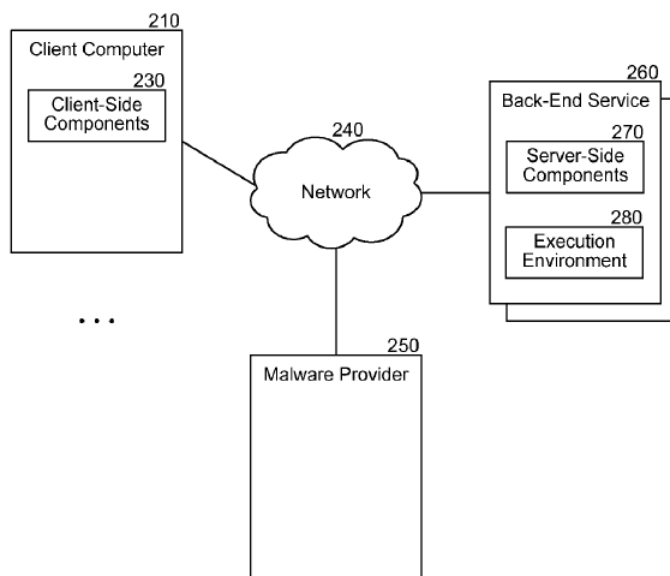


Figure 2

Id., Fig. 2. Here, “[o]ne or more client computers 210 are connected by a network 240 to a back-end service 260.” *Id.*, [0026]. Client computers 210 include “client-

side components 230 of the malware analysis system that detect and analyze information about threats on the client computer 210.” *Id.* This includes “meta-data from the infected computer,” (*see id.*, [0029]) including “historical and hierarchical data that indicates how a potentially malicious piece of code initiated each action” such as the “order of these events and the relationships between them” (*see id.*, [0015])[.]” *See also id.*, [0028]-[0036] (describing client-side data collection).

138. This information is then sent to the back-end service 260 “for further analysis.” *Id.*, [0026]. This analysis may include “reproducing the execution and behavior of detected threats” such that the back-end service can “observe[] the malware execution in the execution environment 280 and provide[] signatures and mitigation information[.]” *Id.*, [0027], [0037]-[0038] (describing functions of back-end service).

139. *Polyakov* further discloses that in embodiments, “the malware analysis system ranks threats received by priority.” *Id.*, [0039]; *see also id.*, [0021] (discussing prioritizing “threat reports based on certain criteria[,]” including “threats that match a particular known threat family” or by an “increasing threat detection rate”). This priority functionality “promotes more timely inspection by technicians or automated analysis, thereby more effectively utilizing limited analysis resources.” *Id.*, [0040]. *Polyakov* explains further that prioritization of threats “improves threat

detection on an on-going basis by focus-ing threat analysis resources on the most troublesome threats.” *Id.*, [0021].

140. Therefore, there would have been a reasonable expectation of success modifying *Morris*’s key to include event priority information, as was well-known in the art as discussed above.

4. *Claim 4 – The method of claim 1, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.*

141. See above, ¶115 (describing *Morris*’s disclosure of “Event” information comprising “three principal components: the key of the object performing the act (the ‘**Actor**’), the act being performed (the ‘**Event Type**’), and the key of the object or identity of another entity upon which the act is being performed (the ‘**Victim**’).”

5. *Claim 8*

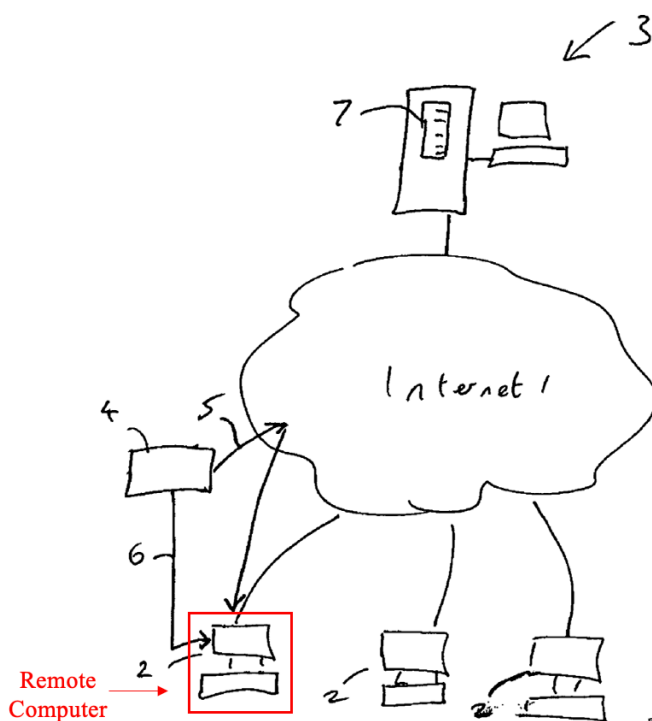
- (a) *8[P]. A system for providing forensic visibility into a system, comprising:*

142. *Morris* discloses an “**apparatus** for dealing with malware.” *Morris*, [0001], Fig. 1. *Morris*’s system provides forensic visibility into a system by analyzing “relationship chain[s]” for events, similar to the audit trails analyzed by the ’045 Patent. See, e.g., *Morris*, [0052], [0099] (“relationship chain provides a complex series of correlation points, allowing ancestral relationships to be

considered for any object and its predecessors”), [0128]-[0135]; *see also* ’224 Patent, 1:18-20 (“systems for providing forensic visibility into client systems having full audit trails are described”).

(b) [8(a)] a device including:

143. *Morris*’s “apparatus for dealing with malware” is depicted below in Figure 1 and includes a *device*, such as a “**remote computer**” boxed in red:



Morris, Fig. 1. For simplicity, I have boxed only one of the plurality of remote computers 2. Remote computer 2 is used to to gather event information and send this information to a base computer for analysis. *Morris*, [0014]-[0016], [0077]-[0078].

(c) [8(b)] a communication interface;

144. *Morris* explains that its remote computers are connected to the “**Internet 1**” or other type of network (e.g., LAN) to enable communication with base computer 3. *Morris*, [0077]. A skilled artisan would have understood *Morris*’s remote computer includes a *communication interface* to facilitate communication across the network to base computer 3. Indeed, *Morris* discloses its remote computers 2 “may each be variously a personal computer, a server of any type, a PDA, mobile phone, an interactive television, or any other device capable of loading and operating computer objects” which a skilled artisan would have understood each to include a communication interface.

(d) [8(c)] a processor;

145. *Morris* further explains that its remote computer 2—which optionally may be a personal computer—is configured to run “local ‘agent’ software” for performing the functions described in its disclosures. *Morris*, [0079]; *see also id.*, [0033] (“This method, which may be carried out by so-called agent software running on the remote computer, allows for efficient sending of data to the base computer...”), [0065] (“the method, which may be implemented in agent software running on the local computer...”). A skilled artisan would have understood remote computer 2 would include a processor for executing the functions of the “local ‘agent’ software.” It would have been commonplace to include a processor (e.g., CPU) in the remote computer to run the software functions.

(e) [8(d)] *data storage; and*

146. As I explained in my opinions above, *Morris* discloses its' methods are effectuated by "agent software **running on the remote computer.**" *Id.*, [0033], [0079], [0122] ("computer 2 may have agent software **installed...**"). One of skill in the art would have understood remote computer 2 thus includes *data storage* which stores this local agent software that is installed for running the processes described therein.

(f) [8(e)] *a sensor agent stored on the data storage that is executable by the processor, wherein the sensor agent is operable to:*

147. As I discussed above, remote computer 2 is configured with "local 'agent' software" for effectuating the described methods and a processor for executing the functions of the "local 'agent' software." *See above*, ¶56, ¶145.

(g) [8(e)(i)-(v)] gather one or more events defining an action of a first object acting on a target; generate a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object; obtain a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information, wherein the global perspective for

one or more related events to the at least one event across a network; assemble an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmit the assembled event line utilizing the communication interface.

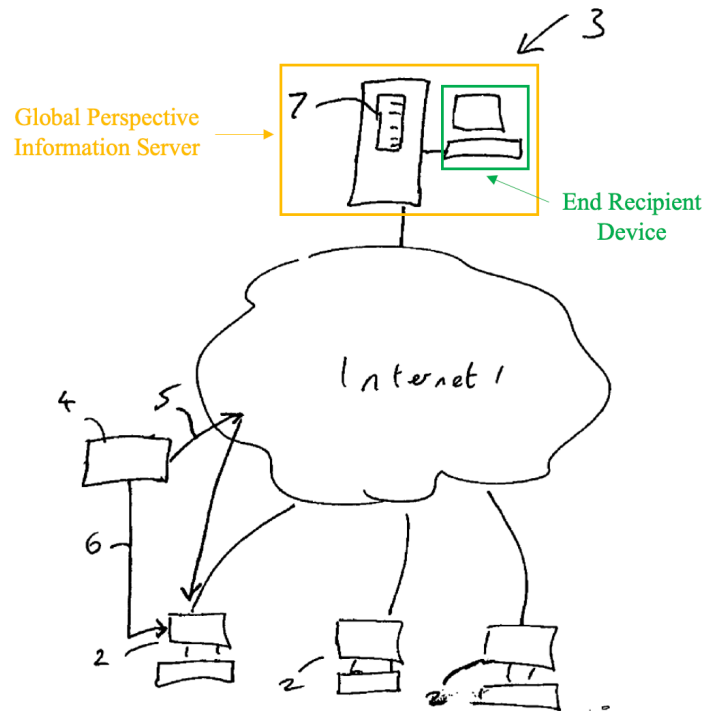
148. I note that limitations 8(e)(i)-8(e)(v) are identical in substance to limitations [1(a)-1(e)] above. These limitations of claim 8 are obvious for the same reasons described in my opinions therein.

6. Claim 9 – The system of claim 8, wherein the at least one of the one or more events is executed by the processor.

149. As I discussed above, *Morris*'s remote computer 2 is configured with “local ‘agent’ software” for effectuating the described methods—including executing the described events—and a processor for executing the functions of the “local ‘agent’ software.” *See above*, ¶56, ¶145. Thus, *Morris*'s processor executes the disclosed events.

7. Claim 10 – The system of claim 8, further comprising: a global perspective information server; and an end recipient device, wherein the information associated with the one or more of the first object and the originating object is obtained from the global perspective information server and the sensor agent transmits the event line to the end recipient device.

150. As I explained in my opinions above concerning claim 1, *Morris* discloses base computer 3 connected to database 7, which together facilitate the comparison and classification of event data. *Morris*, [0014]-[0017], [0020], [0078]. These components are illustrated below:



Id., Fig. 1. A skilled artisan would have understood that base computer 3 and database 7 together function consistently as the claimed *global perspective information server*. I have boxed this global perspective information server in orange above. Similarly, a skilled artisan would have understood that base computer 3 is an *end recipient*, boxed in green.

151. More particularly, *Morris's global perspective information server* is consistent with the disclosures of the '045 Patent. The '045 Patent states “[t]he global perspective information server 216 may comprise a general purpose computer and/or server or other device or federated group of devices capable of supporting communications over a communication network 212.” ’045 Patent, 14:40-44. In *Morris*, base computer 3 and database 7 are a federated group of devices capable of

supporting communications over a communication network which interoperate with remote computer 2, as described above, to compare and classify event data. Base computer 3 and database 7 process requests (i.e., function as a server) from the local computers for authorization to run a file. *See, e.g., Morris*, [0007] (describing for prior art systems, a request for authorization to run the file that is sent by a local computer to the central computer may comprise sending a checksum or “signature” or “key” that uniquely represents the file). In this same way, *Morris*’s system requests authorization to run files from base computer 3/database 7.

152. *Morris* also discloses a malware determination (i.e., *information associated with the one or more of the first object and the originating object*) is obtained by base computer 3 performing an analysis on the data stored in database 7. Illustrated above in Figure 1, “base computer 3 holds a database 7 with which the remote computer 2 can interact when the remote computers 2 run an object 4 to determine whether the object 4 is safe or unsafe.” *Morris*, [0078]. This “database 7 is populated, over time, with information relating to each object run on all of the connected remote computers 2.” *Id.* Base computer 3 compares “the data about the computer object received from the plural computers” and “classif[ies] the computer object as malware on the basis of said comparison.” *Id.*, [0016], [0017], [0081]-[0084] (describing process for determining object is “bad or unsafe”). Thus, the malware determination is obtained from base computer 3 and database 7 (i.e., *global*

perspective information server). Indeed, *Morris* explains “[t]he centralisation of the community data in the community database 7 provides the ability to rapidly assimilate object behaviours, allowing for the rapid identification and determination of malware.” *Id.*, [0100].

153. Furthermore, as I described in claim 1(e), *Morris* discloses its “agent software” running on remote computer 2 (i.e., *sensor agent*) transmits the key comprising the event and contextual data (i.e., *event line*) to base computer 3 (i.e., *end recipient device*) to perform the above-described malware analysis. *See above*, ¶117. And this key can also be copied back to a local computer. *Id.* I note in this process, the local computer is an *end recipient*.

8. *Claim 12 – The system of claim 8, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.*

154. *See above*, ¶115, ¶141 (describing *Morris*’s disclosure of “Event” information comprising “three principal components: the key of the object performing the act (the ‘**Actor**’), the act being performed (the ‘**Event Type**’), and the key of the object or identity of another entity upon which the act is being performed (the ‘**Victim**’).”

9. *Claim 16*

- (a) *16[P]-[16(e)] A system for providing forensic visibility into a system, comprising: a device including: a*

communication interface; a processor; data storage; and a sensor agent stored on the data storage that is executable by the processor, wherein the sensor agent is operable to:

155. I note that limitations 16[P]-16(e) are identical in substance to limitations 8[P]-8(e) above. These limitations of claim 16 are obvious for the same reasons described in my opinions therein.

- (b) *[16(f)]-16(g)(i) gather one or more events defining an action of a first object acting on a target; generate a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object;*

156. I note that limitations 16(f)-16(g)(i) are identical in substance to limitations 1(a)-1(b)(i) above. These system limitations of claim 16 are obvious for the same reasons described in my opinions therein.

- (c) *[16(h)] assemble an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, the originating object, and global perspective information comprising information about one or more related events to the one or more events across a network; and*

157. As I described above in my opinions concerning limitations 1(d)-1(e), *Morris* discloses assembling a key, which I note constitutes the claimed *event line*, including information identifying the first object (the “Actor”), action of the first object (the “Event type”), target (the “Victim”), and originating object (the

“branching information” or multiple sources/pairs of sources) and transmitting this key to a base computer for analysis. *See above*, ¶¶114-117. At the base computer, this analysis includes determining an IP address associated with the first/originating object, determining if the first object and/or originating object is malware, and determining a popularity associated with these objects (i.e., *global perspective information comprising information about one or more related events to the one or more events across a network*). *See above*, ¶¶100-109. And *Morris* also discloses this key at the base computer can then be “copied to” the local database of the remote computer that first ran the object. *Morris*, [0084].

158. A skilled artisan would have understood the key copied back to the remote computer would include the same event information transmitted to the base computer/local database. This includes the IP address of the Victim (i.e., the *global perspective*). Further, it would have been obvious to one of skill in the art to store the malware determinations in this key as well to better inform a user regarding the significance of the event on the computing system. Indeed, identifying that a particular file, event, or originating process is malware would provide additional information to a user to decide on the proper remediation tactics for any malware-based event.

- (d) *[16(i)] transmit the assembled event line utilizing the communication interface.*

159. See above, ¶117. I note this transmission occurs over the Internet, such that the local and base computers would utilize their respective communication interfaces to communicate over this network.

H. Opinions Regarding Ground 2: Claims 1-2, 4, 7-10, 12, and 15-18 are rendered obvious by Capalik or alternatively, Claims 1-4, 7-10, 12, and 15-18 are rendered obvious by Capalik in combination with Van Oorschot

1. Claim 1

(a) *1[P]. A method comprising:*

160. *Capalik* describes “**methods** for protecting computer networks” including “**methods** for analyzing malicious activities on a computer system in order to better protect the computer system from future malicious activity.” *Capalik*, [0004]. This method is generally described below:

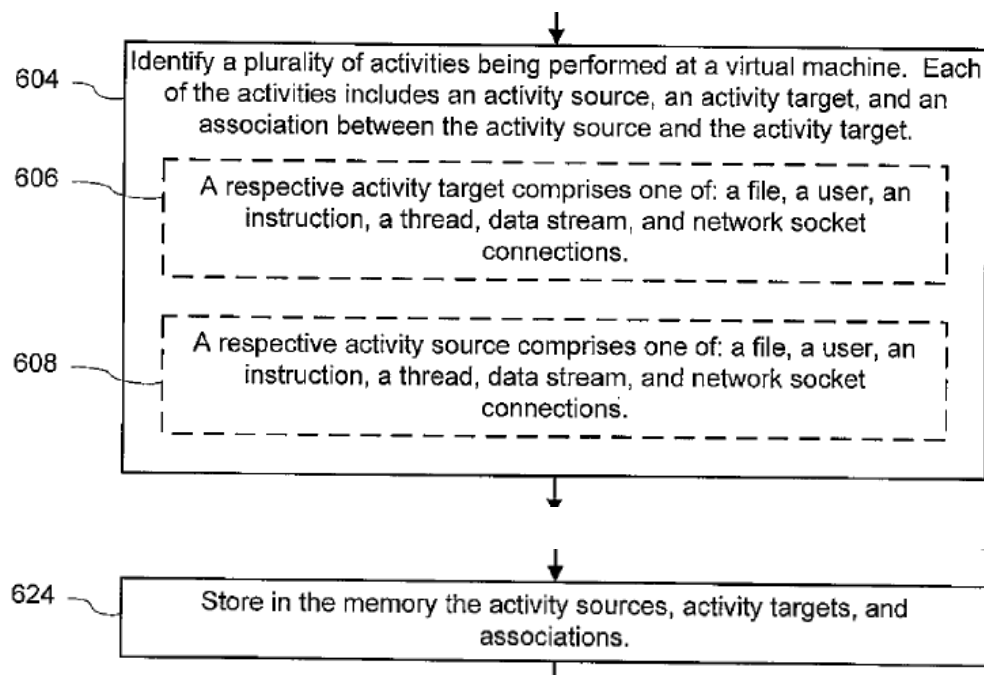
A computer implemented method includes monitoring activity on the virtual machine. A plurality of activities being performed at the virtual machine is identified. Each of the activities includes an activity source, an activity target, and an association between the activity source and the activity target. The activity sources, activity targets, and associations are stored in the memory. A fingerprint indicative of the activity on the virtual machine is created from the stored activities. The fingerprint is transmitted to prevent future attacks that comprise the same or similar activities as indicated by the fingerprint.

Id., Abstract. *Capalik*’s methods of identifying unauthorized activities on a computer system are generally depicted in the flowcharts of Figures 6A-6C. *Id.*, Figs. 6A-6C.

(b) *[1(a)] gathering one or more events defining an action of a first object acting on a target;*

161. As I described above, *Capalik's* methods include “monitoring activity on the virtual machine” where “[e]ach of the activities includes an activity source, an activity target, and an association between the activity source and the activity target.” *Capalik*, Abstract; *see also id.*, [0010]. I note that *Capalik's* “activities” constitute “events” as claimed. These activities are captured as “raw attack information” and stored in “database server” as forensic data for later analysis. *Id.*, [0063]; *see also id.*, [0010] (disclosing the “activity sources, activity targets, and associations are stored in [] memory”).

162. *Capalik's* process for gathering activities (i.e., *events*) including a source (i.e., *first object*) and an activity target (i.e., *target*) is illustrated in the excerpted flow diagram below, taken from Figures 6A and 6B:



Id., Figs. 6A and 6B (excerpts).

163. Below I provide an exemplary event gathered by *Capalik*. Figure 3D below illustrates an “exemplary network diagram[] representing unauthorized activity on a computer system (e.g., the virtual machine 113-1)[.]” *Id.*, [0067]. There, a data Packet Stream 304 at a Socket 306 is monitored by the virtual machine. *Id.*, [0068]. Packet Stream 304 is directed to a Process A 308, which accesses File X 310, and File X 310 then initiates a Process B. *Id.*, [0069]. This example event is illustrated below:

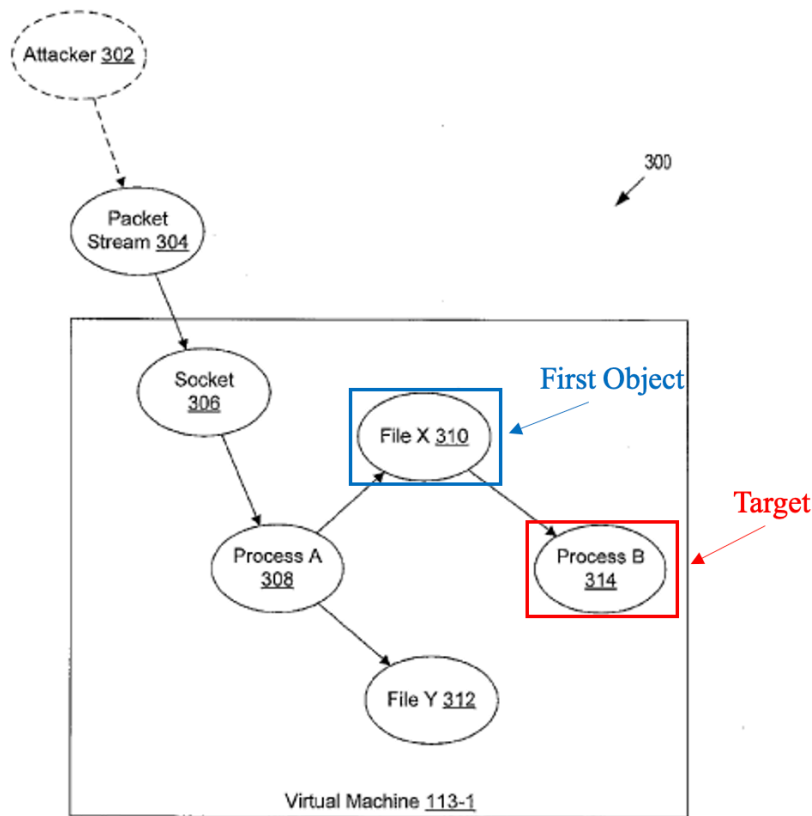


Figure 3D

Id., Fig. 3D. Notably, this diagram represents several related “activities” or “events”:

(1) sending Packet Stream 304 into the computer system through Socket 306; (2)

directing Packet Stream 304 to Process A; (3) Process A accessing Files X and Y; (4) File X initiating a Process B. For simplicity, I have only annotated the final activity. In this activity, File X boxed in blue (i.e., *first object*) performs an action of initiating a Process B boxed in red (i.e., *target*). I note, however, that each of the four activities shown above comprises its own *first object* and *target*. Indeed, *Capalik* itself explains that Process A can be the activity target of one activity and the activity source of another activity. *Capalik*, [0115] (“For example, FIG. 3C illustrates the first activity between the socket 306 and the process A 308 and the second activity between the process A 308 and the file X 310. The process A 308 is the activity target of the first activity and the activity source of the second activity”).

164. *Capalik*'s system gathers unauthorized activity data from these events, and stores this data in data structures. For example, the following is an exemplary data structure for storing unauthorized activity data:

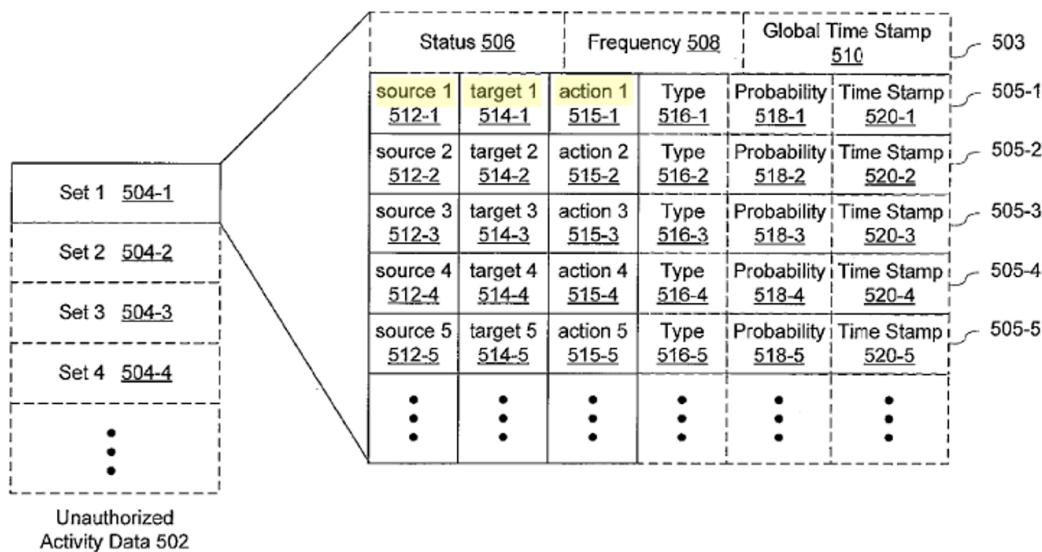


Figure 5A

Id., Fig. 5A. “Unauthorized activity data 502 includes unauthorized activity entries 505,” comprising: “an activity source 512, which calls, accesses, or executes the activity target 514;” “an activity target 514, which is called, accessed, or executed by the activity source 512;” and “an action 515,” which represents an “operation occurring between the activity source 512 and the activity target 514,” each of which I have highlighted above. *Id.*, [0075], [0081]-[0083]. Each entry can include “one or more” of a source, target, action, etc. *Id.*, [0080].

165. It is my opinion, therefore, that *Capalik* discloses *gathering one or more events defining an action of a first object acting on a target*.

- (c) *[1(b)] generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object,*

166. As I described above, *Capalik* discloses that its methods include gathering event data, including a first object acting on a target, such as the objects and targets illustrated in Figure 3D, reproduced below:

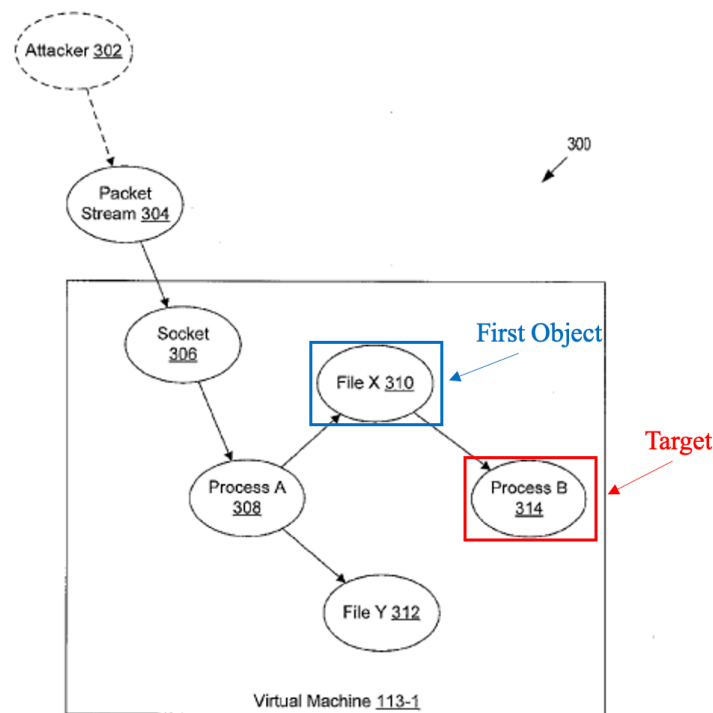


Figure 3D

Id., Fig. 3D. *Capalik* explains that once unauthorized activities are detected, “**related unauthorized activities are identified.**” *Capalik*, [0069]. In Figure 3D above, if “packet stream 304 and the socket 306 are identified as affected by unauthorized activities,” the chain of related events is identified. *Id.*, [0069]; *see also id.*, [0130] (describing *Capalik*’s methods generally providing the ability “to follow a chain of

unauthorized activity”). As part of this chain of related events, *Capalik* discloses correlating an event to an originating object.

167. Particularly, this “originating object” is depicted below in Figure 3D:

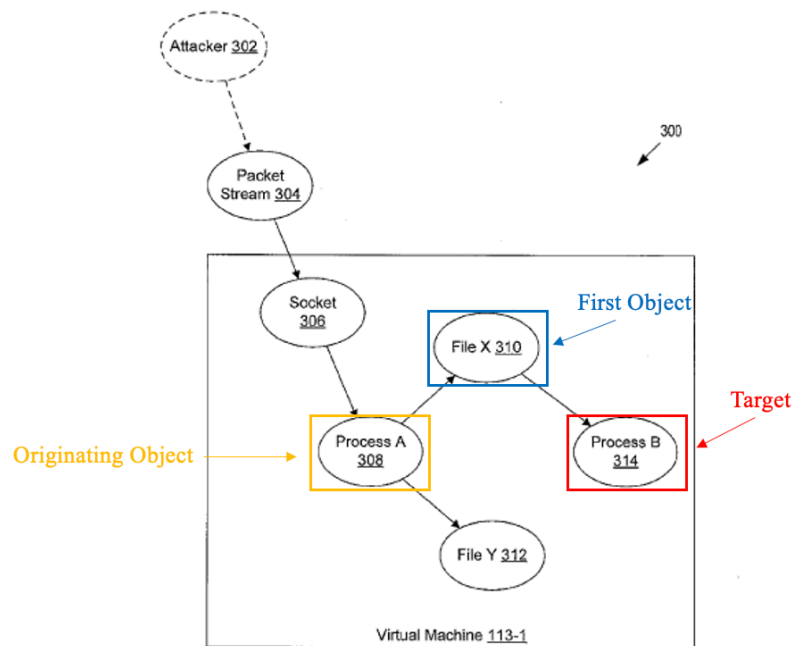


Figure 3D

Id., Fig. 3D. I boxed the originating object “Process A” in orange above. In this sequence, File X 310 initiates Process B 314. Process A 308 serves as the originating object—the process which accesses file X 310 in the related activity chain. I note that *Capalik*’s disclosure of correlating an event back to Process A is consistent with the ’045 Patent’s disclosures. For example, the ’045 Patent defines an object broadly as a “computer file, part of a file or a sub-program, macro, web page or any other piece of code to be operated by or on the computer, or any other event whether executed, emulated, simulated or interpreted.” ’045 Patent, 7:32-36. This definition

includes an object as a process. *See also, id.*, 3:45-47 (describing an “originating process”), 17:39-50 (noting “Object 2” is “a process”). Furthermore, the ’045 Patent explains, like in *Capalik*, that an “originating process or object may be determined based on a previous event that initiated or otherwise created the current process and/or object.” *Id.*, 17:39-42. Thus, *Capalik*’s disclosure of correlating an event to Process A discloses *correlating the at least one event to an originating object*.

168. For example, *Capalik* explains “the unauthorized activities start with the packet stream 304; the unauthorized activities send the packet stream 304 into the computer system [] through the socket 306; the packet stream 304 is directed to the process A 308[.]” *Id.*, [0070]. Process A “is treated as affected by unauthorized activities...[and then] process A 308 accesses a file X 310 and a file Y 312, and both the file X 310 and the file Y 312 are treated as affected by unauthorized activities.” *Id.*, [0069]. “Thereafter, the file X 310 initiates a process B 314, and the process B is treated as affected by unauthorized activities.” *Id.* In light of *Capalik*’s discussion here, a skilled artisan would have understood that *Capalik* correlates each event to its’ originating object by tracking the related chain of unauthorized activities.

169. Therefore, in my opinion, *Capalik* discloses *generating a contextual state for at least one of the one or more events by correlating the at least one event to an originating object*.

- (d) [1(b)(i)] the contextual state including an indication of the originating object of the first object and an indication

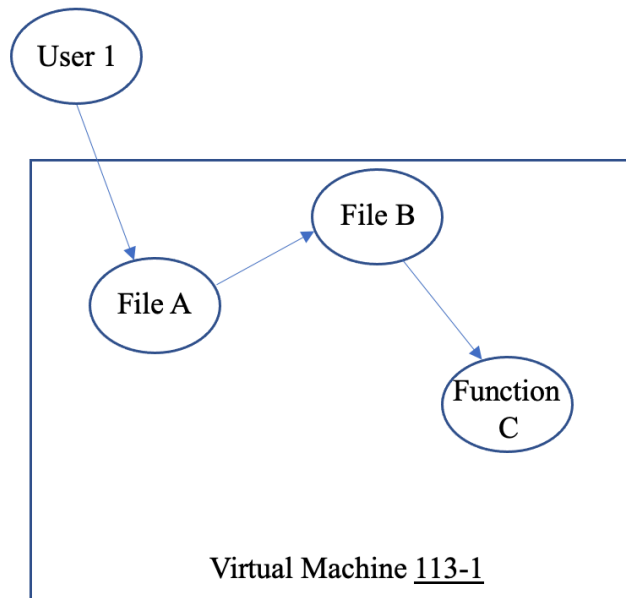
of at least one of a device on which the first object is executed and a user associated with the first object;

170. I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “at least one of” a “device on which the first object is executed and a user associated with the first object[.]” In other words, to render this limitation unpatentable, the prior art need only disclose one of the two: the device where the first object is executed or a user associated with the first object.

171. As I explained above in the preceding limitation, *Capalik* discloses generating a contextual state for an event by correlating an event to originating object of the first object. The originating object is boxed above in Figure 3D in orange. The first object is boxed above in Figure 3D in blue. *Capalik* additionally discloses that this contextual state can include a device where the first object is executed and/or a user associated with the first object. For example, with respect to a device where the first object is executed, *Capalik* discloses that each of the activity source and the activity target may be “an **Internet Protocol address**” (i.e., *a device on which the first object is executed*). *Capalik*, [0104]-[0105].

172. With respect to a user associated with the first object, *Capalik* explains that “a respective activity source comprises [] one of: a file, **a user**, an instruction (such as computer processes or system calls), a thread (e.g., a thread of processes),

data stream (e.g., network packets), and network socket connections (or sockets)” *Id.*, [0104]; *see also id.*, [0105] (explaining a target may also be “a user”). Take the following simple example of an event I have constructed for occurrence on *Capalik’s* Virtual Machine 113-1:



In this event, User 1 downloads File A which accesses File B to execute programming Function C. The first object (File A) is downloaded by the user. As part of the related chain of unauthorized activity, *Capalik’s* methods identify the originating source of this activity, which is User 1 who downloaded File A. User 1, therefore, is the user associated with the first object (File A)—the source which downloaded this file.

173. This same rationale applies when a target is a device at a certain IP address. In a related example, a user (User 1) downloads a file (File A) on a device

(IP Address 1). As before, File A is the first object, and File A is executed on the target device having IP Address 1. *Id.* Thus, the device at IP Address 1 is the device on which the first object is executed. *Id.*

174. Should Patent Owner or its expert suggest that *Capalik* does not disclose *an indication of at least one of a device on which the first object is executed and a user associated with the first object*, *Van Oorschot* does as I described in my opinions above at ¶¶93-98. In particular, *Van Oorschot* describes a similar computer security system comprising an “event management tool” used to normalize event data including “‘who’ (the user initiating the event)” and “‘where’ (the system or device on which the event occurred).” This event information is depicted below, Figure 5A illustrating event information pre-normalization and Figure 5B illustrating normalized event data:

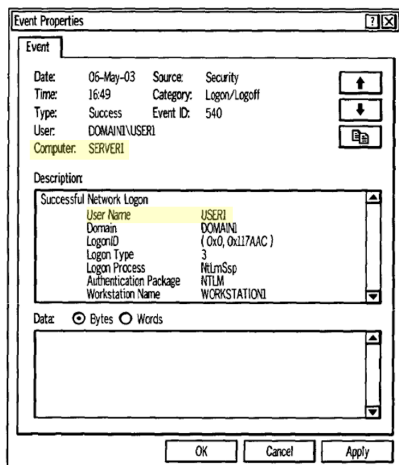


FIG. 5A

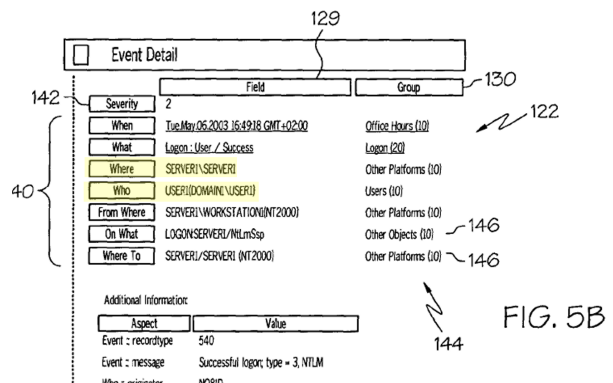


FIG. 5B

Van Oorschot, Figs. 5A and 5B.

175. One of skill in the art would have found it obvious to modify *Capalik's* system to generate additional types of activity data, including “where” (i.e., “the device where the event occurred”) and “who” (i.e., “the user initiating the event”). This combination adheres to recognized computer security principles, such as gathering more detailed threat information provides improved threat analysis and detection where limited data acquisition can hinder back-end performance and result in unperceived threats. One of skill in the art would have been motivated to collect device and user information associated with an event to predictably improve the quality and success of *Capalik's* back-end threat identification and analysis. Such information is both common and readily available and in the context of threat detection, is particularly useful. For instance, a user or security administrator may want to identify the user or device associated with a threat to identify the initial source of malware to “blacklist” this user or device on the network to prevent future threats. Simply identifying the root “bad actor” provides a user with a variety of remediation tactics.

176. A skilled artisan would also have been motivated to make this modification to *Capalik* because *Capalik* explains that “additional...data structures not described above” related to activity data can be stored in memory for fingerprint generation and backend analysis. *Capalik*, [0054]; *see also id.*, [0120] (disclosing “collecting additional data associated with the activity sources” to create a

fingerprint from this data). Based on *Capalik's* disclosures here, a skilled artisan would have been expressly motivated to store additional data, such as the device where the event occurred and the user initiating the event, as described by *Van Oorschot*.

177. A skilled artisan would have understood this combination of prior art elements would have been accomplished using known methods, such as simply collecting additional known event information, with no change to their respective functions. *Capalik's* system would derive additional activity data in addition to the disclosed data in accordance with its current event capturing functions. Thus, one of skill in the art would have appreciated a reasonable expectation of success making this simple modification.

- (e) *[1(c)] obtaining a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information,*

178. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “the information including at least one of” an “age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware,

Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information[.]” To render this limitation unpatentable, the prior art need only disclose one of the items in the above list.

the information including at least one of...Internet Protocol (IP) Address

179. As I explained in my opinions above, *Capalik* discloses obtaining the “**Internet Protocol address**” of the device where the first object is executed. *See above*, ¶171. In the example event I provided above, User 1 downloads File A onto a device identified as IP Address 1. There, the first object, File A, is executed on the device at IP Address 1. In my opinion, obtaining the IP Address of the device where Object 1 is executed discloses the claimed *obtaining a global perspective for the at least one event by obtaining information associated with...the first object...the information including at least...Internet Protocol (IP Address)*.

the information including at least one of... a determination as to whether the first object is malware, a determination as to whether the originating object is malware

180. *Capalik* additionally explains that its method includes determining whether the first and the originating objects are malware. For example, *Capalik* explains that its method includes determining “association types” between sources and targets, such types including “**malicious associations** v. benign associations.” *Capalik*, [0075]. These “association types” are stored in the following exemplary data structure, along with other event information:

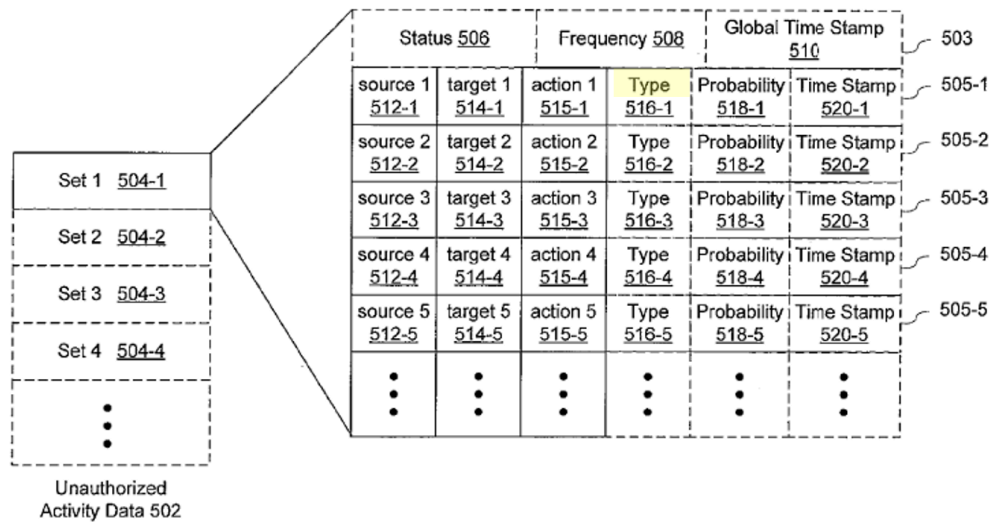


Figure 5A

Id., Fig. 5A. I have highlighted the “Type” 516-1 which indicates the “association types” discussed above, such as malicious or benign.

181. *Capalik* provides that “[e]ach set 504 in the unauthorized activity data 502 in Fig. 5A [] includes one or more unauthorized activity entries” which comprises, in relevant part, “an activity source 512[;]” “an activity target 514[;]” “an action, 515, which represents an...operation occurring between the activity source 512 and the activity target[;]” and optionally an “**association, type 516**, which when identified, represents the type of association between the activity source 512 and the activity target 514.” *Id.*, [0080]-[0084]. This “type” may be identified as “**malignant**, which represents that the activity source 512 has modified the activity target 514 to perform unauthorized activities.” *Id.*, [0084]-[0085].

182. More particularly, *Capalik* discloses determining malicious or malignant associations between sources and targets in the events occurring on virtual machine 113-1, as follows:

“as shown in FIG. 3C, the process A 308 modifies the file X 310, performing unauthorized activities (e.g., **corrupting data or writing a malicious code**). **The process A 308, as an activity source to the file X 310, causes unauthorized activities on the file X 310.** By identifying that the process A 308 performed unauthorized activities, **the process A 308 is determined to be affected by the unauthorized activities, and the association between the socket 306 and the process A 308 is determined to be malicious** (i.e., unauthorized).”

Id., [0106].

183. One of skill in the art would have understood determining Process A to be “malicious” by, for example, writing “**malicious code**,” includes determining Process A (i.e., *first object*) to be malware. Similarly, tracking back the related chain of activity to socket 306 to determine a malicious association between socket 306 and Process A likewise includes determining socket 306 (i.e., *originating object*) to be malware. I note that this is consistent with the ’045 Patent’s definition for “malware,” which “refer[s] generally to any executable computer file or, more generally ‘object’, that is or contains **malicious code**...” ’045 Patent, 2:1-5.

184. In my opinion, therefore, *Capalik* discloses obtaining a global perspective for the event by obtaining information concerning *a determination as to whether the first object/originating object is malware*.

- (f) *[1(c)(i)] wherein the global perspective for one or more related events to the at least one event across a network;*

185. For the purposes of my analysis, I have been asked to interpret this limitation to require the global perspective include “information about events across the network” where those events occur on “other devices” on the network.

186. As I explained above, *Capalik* discloses obtaining a global perspective for an event, including obtaining an IP Address of a first object and making a malware determination for a first object and/or originating object. *Capalik* further discloses that its methods “need not be restricted to analysis of a single virtualized operating system, but, rather, **are able to follow a chain of unauthorized activity across multiple virtualized operating systems.**” *Capalik*, [0130]. This enables “identifying and thwarting advanced threats that involve multiple attackers spread across multiple systems[.]” *Id.*

187. One of skill in the art would have understood that in an attack across multiple virtualized operating systems, *Capalik*’s methods would follow the related chain of unauthorized activity across each operating system and would perform the methods identified above for obtaining a global perspective—namely, identifying an activity source and the activity target (e.g., by “an Internet Protocol address”) and determining “association types” between sources and targets, including “malicious associations.” By following an attack across multiple virtualized operating systems and gathering, generating, and obtaining the information discussed above, *Capalik*’s

disclosed methods include obtaining a global perspective for an event where such global perspective includes information about events across the network. This global perspective includes obtaining an IP Address for the first object or a malware determination for the first object or originating object for an attack spread across multiple systems.

188. Should Patent Owner suggest that *Capalik's* multiple virtualized operating systems are limited to a single device, a skilled artisan would have understood *Capalik's* system is scalable and would have been motivated to implement multiple virtualized operating systems not only within a single computer, but also across a plurality of decoy network devices to protect network devices 136-n. Indeed, *Capalik* already contemplates implementing multiple decoy network devices in its system and methods. *See, e.g., Capalik*, [0030] (“The decoy computer network 102 includes **one or more decoy network device(s)**”). In fact, it was well known in the art to accumulate information from several attacked devices to inform a malware determination. *See above*, Claim 1 (*Morris* mapping). It would have been obvious to scale *Capalik's* system according to known methods to implement multiple decoy network devices as contemplated by *Capalik*, each comprising their own set of virtual machines/virtual operating systems. There, when an attack spreads across multiple systems (i.e., multiple decoy network devices), the associated activity chain would be tracked and a malware determination would be made based

on the monitored information. *See, Capalik, [0130].* Implementing multiple decoy network devices each with their own set of virtual machines/virtual operating systems would enhance *Capalik's* methods to “identify[] and thwart[] advanced threats that involve multiple attackers spread across multiple systems.” *Id.* For example, a multiplicity of decoy network devices would increase the amount of data traffic that the system can handle during peak usage and the amount of data in general that may be processed. *See, e.g., Capalik, [0056].*

189. Furthermore, *Capalik* discloses obtaining global perspective information in a scenario based on related events from multiple data streams across a network. This scenario is illustrated below in Figure 4E:

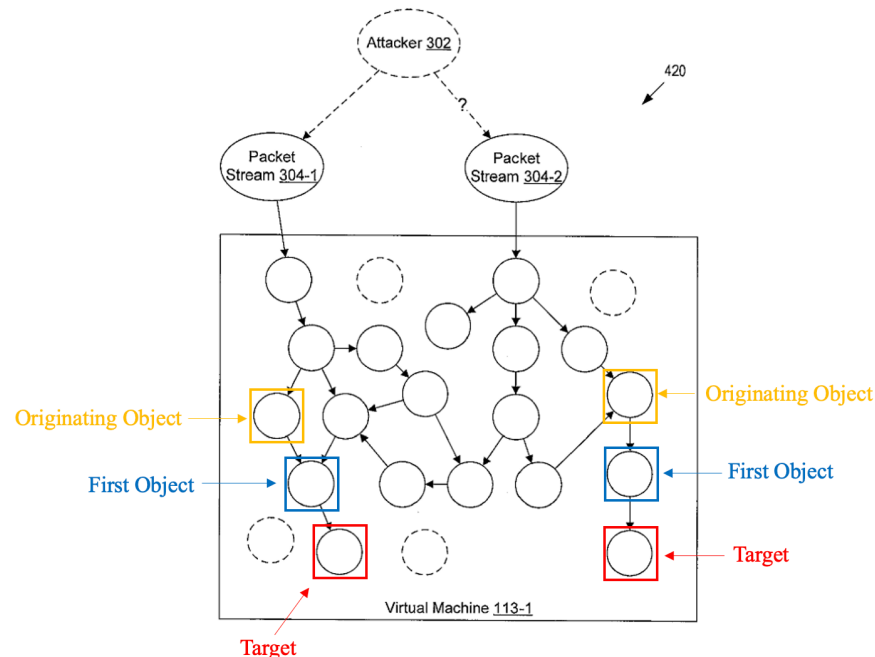


Figure 4E

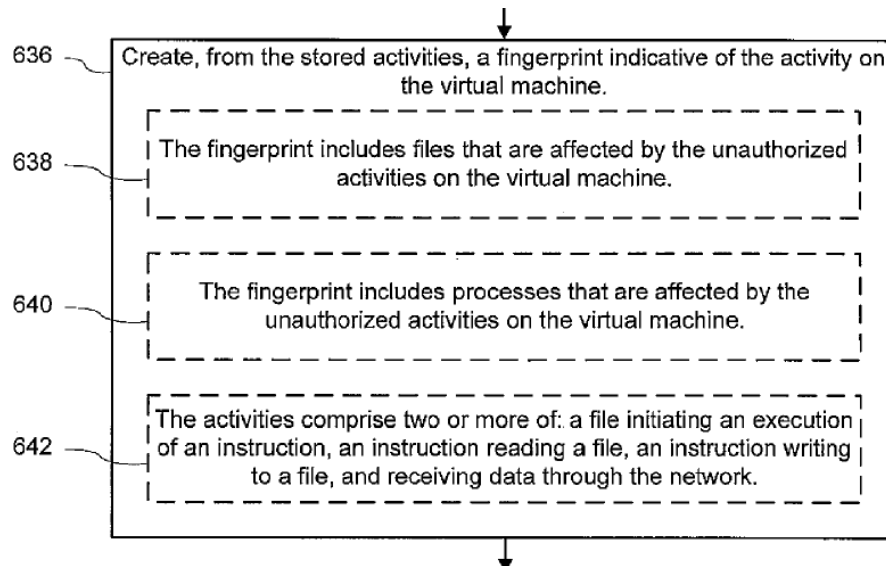
Capalik, Fig. 4E. I have annotated the originating objects (in orange), first objects (in blue) and targets (in red) from each packet stream. While the first object in Packet Stream 304-1 has multiple originating objects, I have identified only one for simplicity. In the above event with multiple data streams, *Capalik*'s methods would similarly obtain the "Internet Protocol address" of the device where the first objects are executed for each data stream and would make a malware determination for the originating and/or first objects. As above, a skilled artisan would have understood that in this scenario, multiple decoy network devices could be employed to track this more sophisticated attack involving several different packet streams—e.g., each network device would be operable to protect a specific port to monitor for unauthorized activities.

- (g) *[1(d)] assembling an event line including details associated with the at least one event, the details including information uniquely identifying the first object, the action of the first object, the target, and the originating object; and*

190. *Capalik* explains that after obtaining the above activity information and associated data, "**[a] fingerprint indicative of the activity on the virtual machine is created from the stored activities.**" *Capalik*, [0010]. *Capalik* provides that this "fingerprint is then transmitted to one or more other computer systems on the network to prevent future attacks that comprise the same or similar activities as

indicated by the fingerprint.” *Id.* This fingerprint can be generated by fingerprint generation engine 130. *Id.*, [0063].

191. *Capalik’s* fingerprint generation process is illustrated below in an excerpt from Fig. 6C:



Id., Fig. 6C. I have included above only the relevant portion of *Capalik’s* flow chart of Figure 6C related to creating a fingerprint from stored activity data. As shown in the above excerpt, *Capalik’s* “fingerprint” includes stored activity data, including the files and/or processes that were affected by unauthorized activities on the virtual machine. *Id.*

192. An example of *Capalik’s* assembled “fingerprint” is illustrated below in Figure 5B:

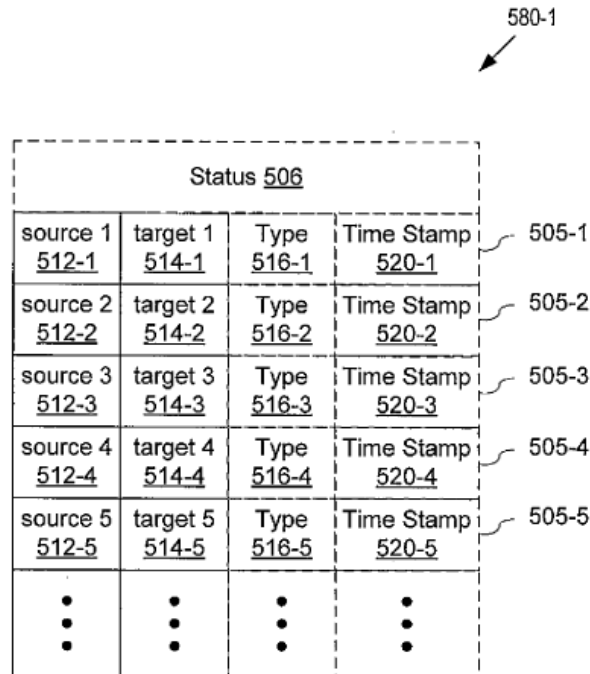


Figure 5B

Id., Fig. 5B. Fingerprint 580 includes “an activity source 512, an activity target 514, an (optional) association type 516, and an (optional) time stamp 520”—i.e., *information uniquely identifying the first object and the target*. *Id.*, [0090]. Capalik also discloses the fingerprint can store “branching information” (see Capalik, [0099]) along with “(one or more) activity sources 512,” including “respective pairs of activity sources...when the association type for a respective pair is malignant.” *Id.*, [0092]-[0094]. A skilled artisan would have understood this “branching information” and/or multiple activity sources and pairs of activity sources would include the *originating object* of the first object.

193. *Capalik* additionally discloses that “fingerprint 580 includes one or more of...raw network information collected from a respective virtual machine 113 or a respective decoy operating system 112[.]” *Capalik*, [0099]. One of skill in the art would have understood this includes storing any of the raw attack information originally stored in memory, such as the attack information identified in Figure 5A:

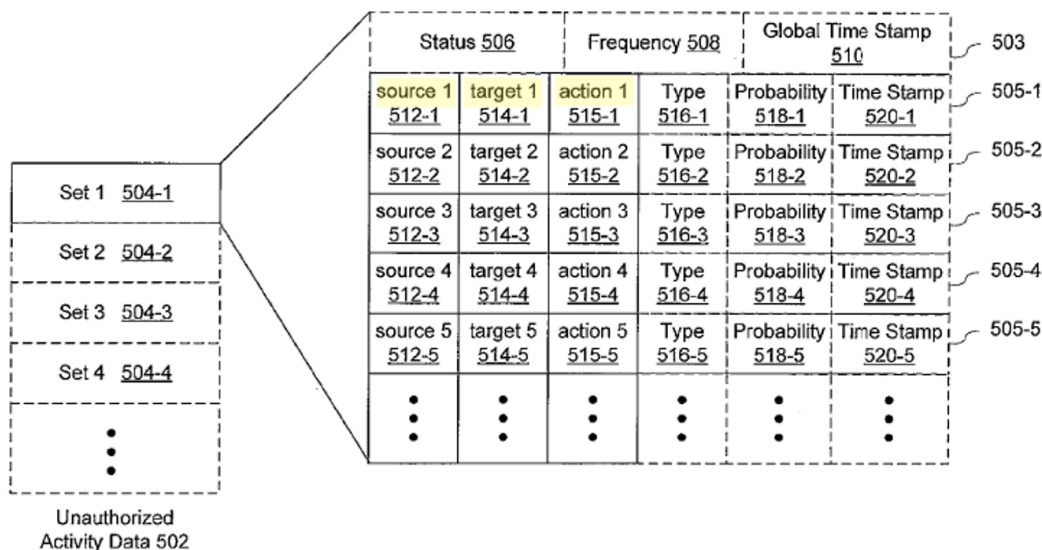


Figure 5A

This includes, for example, the activity source and target discussed above, along with an “**action 515**” which represents an “operation occurring between the activity source 512 and the activity target 514[.]” *Capalik*, [0083].

194. Indeed, *Capalik* provides various embodiments where the fingerprint can store a wide set of different information. *See, e.g., Capalik*, [0121]-[0125]. There, the fingerprint can store files or file information “(e.g., e.g., file name, location, size, checksum value, etc.);” process or process information (“e.g.,

process name, size, privilege level, related processes, etc.)[;]” or can include one or more of:

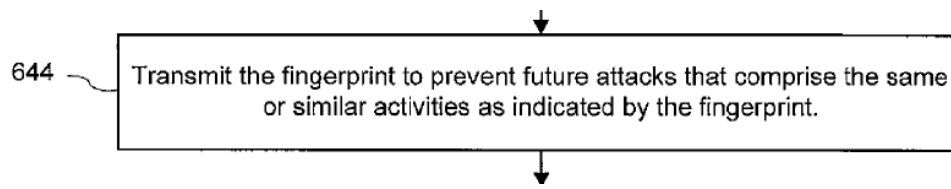
system calls’ arguments and returns, device and memory input-output, driver calls’ arguments and returns, library calls’ arguments and returns, branching information, instruction pointer jumps, and raw network information collected from a respective virtual machine 113 or a respective decoy operating system 112... port numbers, data streams, file uploads, file downloads, keystrokes, ASCII or binary files, malicious payloads, operating system configuration changes (e.g., new or updated registry keys in windows) and memory manipulation attempts.

Id., [0121]-[0124]; *see also id.*, [0125] (describing fingerprint further storing activities comprising “two or more of: a file initiating an execution of an instruction, an instruction reading a file, an instruction writing to a file, and receiving data through the network”). A skilled artisan would have therefore understood that *Capalik’s* fingerprint stores a wide set of information, and such information includes at least the activity source, activity target, the action performed by the activity source, and branching information concerning this activity source which includes the originating object.

195. Therefore, in my opinion, *Capalik* discloses *assembling an event line* (fingerprint 580) *including details associated with the at least one event, the details including information uniquely identifying the first object* (activity source 512), *the action of the first object* (action 515), *the target* (target 514), *and the originating object* (branching information/multiple activity sources).

(h) *[1(e)] transmitting the assembled event line.*

196. As I explained in my opinions concerning the previous limitation, *Capalik* discloses transmitting fingerprint 580 “to one or more other computer systems on the network.” This transmission of *Capalik*’s fingerprint is described below in Figure 6C:



Id., Fig. 6C. As above, I have excerpted the relevant portion of Figure 6C here to emphasize *Capalik*’s transmission disclosures. Aside from transmitting the fingerprint to one or more other computer systems on the network, *Capalik* additionally discloses that it can transmit this fingerprint to “protected network devices 136” or to “the IDS/IPS system 142” to prevent future attacks that comprise the same or similar activities as indicated by the fingerprint. *Id.*, [0126].

2. *Claim 2 – The method of claim 1, wherein the information associated with the one or more of the first object and the originating object is obtained from a community database; and wherein the event line is transmitted to an end recipient.*

197. As explained in my opinions above, *Capalik* discloses obtaining information associated with the first object and/or originating object in three ways: (1) obtaining an IP Address of the device where the first object is executed; (2) obtaining a determination that the first object is malware; and (3) obtaining a

determination that the originating object is malware. *Capalik* additionally discloses that the activity data above is stored in a “**database server 128** as forensic data 236 for later analysis.” *Capalik*, [0063]. Specifically, *Capalik* explains that the decoy network device stores “the activity sources, activity targets, and associations in...database server 128[.]” *Id.*, [0112]. Database server 128 may be stored on decoy network device 106, or “located outside the decoy network device 106.” *Capalik*, [0055].

198. One of skill in the art would have understood database server 128 is a *community database* because it stores “chain[s] of unauthorized activity across multiple virtualized operating systems” or data from “multiple packets of data streams,” which includes unauthorized activity from multiple attackers, multiple devices, or multiple data streams. *Capalik*, [0130]. Indeed, it was known in the art to store information in a “central” community database for comparison with other data across a network. *See above*, Background of the Technology, ¶¶47-48 (discussing *Dozortsev*, *Chess*, etc.). Similar to these community-based malware systems, *Capalik*’s database server 128 stores data from multiple sets of activity across a network. Further, *Capalik* incorporates by reference Application No. 11/488,743 which issued as U.S. Patent No. 8,429,746 to Alen Capalik (Ex. 1024). *Capalik*, [0003]. The ’746 patent here describes a similar signature generation process where these signatures are sent to a “global database of attack signatures[.]”

Ex. 1024, 6:47-49. Thus, it was well-known in the art to store signatures in a community database. In line with this, a skilled artisan would have understood that *Capalik's* database server 128 would constitute a “community database” as was well-known in the art.

199. Furthermore, as I discussed above, *Capalik* further explains that fingerprint 580 can be transmitted to one or more other computer systems on the network, to protected network devices 136, or the IDS/IPS system 142. A skilled artisan would have understood each of these would represent an *end recipient*.

3. *Claim 3 – The method of claim 1, further comprising: obtaining an event priority based on one or more of the generated context and the global perspective for the at least one event, wherein the event line further includes the event priority for the at least one event.*

200. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “obtaining an event priority based on one or more of” the “the generated context and the global perspective for the at least one event[.]” To render this limitation unpatentable, the prior art need only disclose one of these items.

201. *Capalik* discloses processes for determining the probability an event is malicious and for assigning a monitoring priority for malicious activities based on this probability. In this regard, *Capalik* discloses its’ decoy network device can

“determine[] a probability of a respective association” which “represents [the] likelihood that the association is malignant (or benign)...” *Capalik*, [0118]; *see also id.*, [0088]. This probability 518 is shown in Figure 5A. *Id.*, Fig. 5A. *Capalik* explains that this “probability” can be used to adjust a “monitoring priority.” *Id.*, [0119]. *Capalik*’s “monitoring priority” is set based on types of unauthorized activities or certain devices with high frequency of unauthorized activities. *Id.*, [0062].

202. However, while *Capalik* discloses assigning a priority to adjust how events are monitored, *Capalik* does not expressly discuss assigning priority to events which have already been monitored in order to prioritize the order in which they are analyzed (or to provide some other form of event priority post-monitoring).

203. As I described above at ¶¶63-67, *Van Oorschot* discloses “**assign[ing] a severity level**” to “system events” based on the generated context to inform the manner in which an event is to be processed. These severity levels provide an *event priority* for these system events. One of skill in the art would have been motivated to modify *Capalik*’s malware detection methods to incorporate *Van Oorschot*’s assignment of severity levels to appreciate the known in the art benefits discussed above—namely, focusing threat analysis on the most troublesome threats to conserve limited resources. *See, e.g., Polyakov*, [0021], [0040]. Indeed, such priority

weighting processes were well-known in the art. *See above*, ¶¶134-140 (discussing *Hernacki*, *Mashevsky*, and *Polyakov*).

204. A skilled artisan would have appreciated a reasonable expectation of success making this modification. For example, in *Capalik's* modified methods, after gathering event information, generating contextual information and obtaining a global perspective as described above, *Capalik's* system would incorporate *Van Oorschot's* assignment of severity levels. Here, *Capalik's* system would be modified to assign severity levels to threat events based on, for example, whether the threat violates a certain rule outlined in *Van Oorschot* and whether the event attributes obtain a certain score based on the “significance” scoring. Doing so describes a process for *obtaining an event priority based on...the generated context*, as claimed. More particularly, an event priority is obtained by assigning a severity level to an event that is based on the event attributes and/or whether the event violates a pre-proscribed rule. In this way, *Capalik's* methods analyze the contextual information surrounding an event to determine if the present threat presents a substantial risk.

205. It would have been obvious to a skilled artisan to include the event priority in *Capalik's* fingerprint to identify the significance of a particular event. Doing so would provide additional guidance to a user, system administrator, or IDS/IPS system when responding to events. Indeed, as I discussed above in paragraphs 134-140, it was well-known in the art to include priority information in

fingerprints and event logs as discussed in *Van Oorschot*. Accordingly, there would have been a reasonable expectation of success modifying *Capalik*'s signature to include event priority information, according to well-known methods.

4. Claim 4 – The method of claim 1, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.

206. As I explained above, *Capalik* discloses gathering event information, including “unauthorized activity entries 505,” comprising: “an activity source 512, which calls, accesses, or executes the activity target 514;” “an activity target 514, which is called, accessed, or executed by the activity source 512;” and “an action 515,” which represents an “operation occurring between the activity source 512 and the activity target 514.” This event information is reproduced below in Figure 5A:

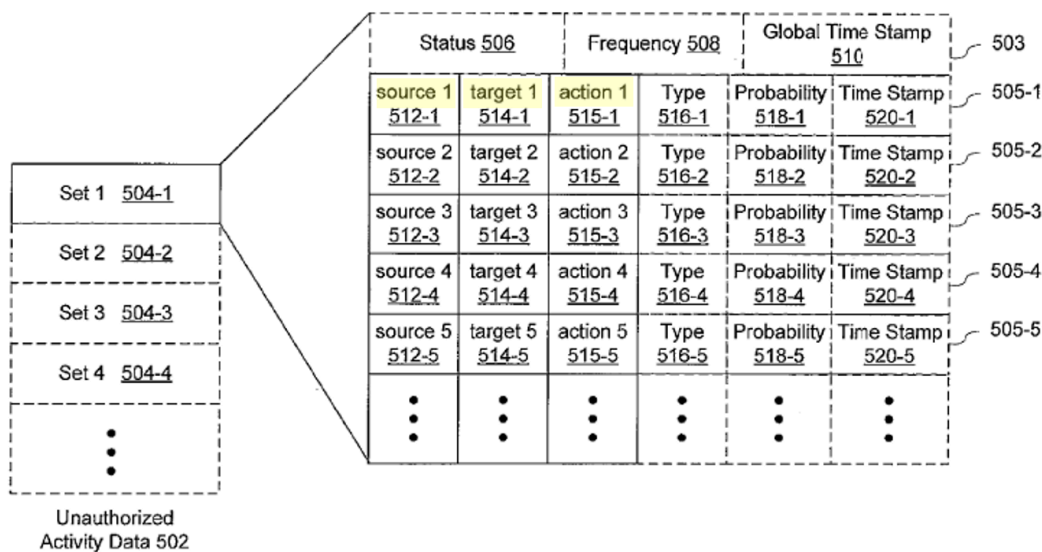


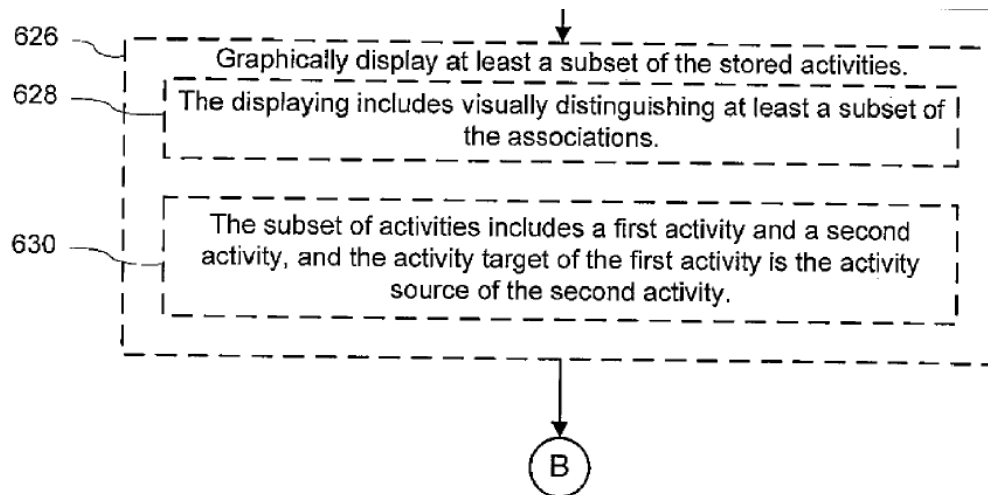
Figure 5A

I have highlighted each relevant set of event information in the figure above.

5. Claim 7 – The method of claim 4, further comprising: heuristically displaying information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.

207. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “the global perspective according to one or more of” “a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.” To render this limitation unpatentable, the prior art need only disclose one of the items in the above list.

208. *Capalik* discloses its methods also include “**displaying** at least a subset of the activities (e.g., **action sources**, **action targets**, and **actions**).” *Capalik*, [0011]. The process for graphically displaying activity data is illustrated below in Figure 6B:



Id., Fig. 6B. I have excerpted the relevant portion of Figure 6B above. *Capalik* thus discloses *heuristically displaying information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed.*

209. *Capalik* further explains that the graphical representation can be “an activity diagram, as shown in FIGS. 3A-3D and 4A-4E[,]” examples of which are provided above. *Id.*, [0113]. For example, with respect to Figure 4E, *Capalik* discloses “[w]hen the type of associations is identified (e.g., malignant, benign, etc., as described in more detail below with reference to a type 516 in FIG. 5A), at least a subset of **the associations can be visually distinguished** (e.g., **a respective bold arrow representing a malignant association...**” *Id.*, [0074]. A skilled artisan would have understood this includes displaying a *global perspective according to...a location associated with a first object*. Here, a source (i.e., *first object*) is determined to be malicious based on its location in a chain of unauthorized activity.

210. More specifically, in annotated Figure 4E below, if the originating object (boxed in orange) is determined to perform unauthorized activities, then the bold arrow pointing to the first object (boxed in blue) visually identifies this “malignant association” such that the first object is similarly identified as performing unauthorized activities:

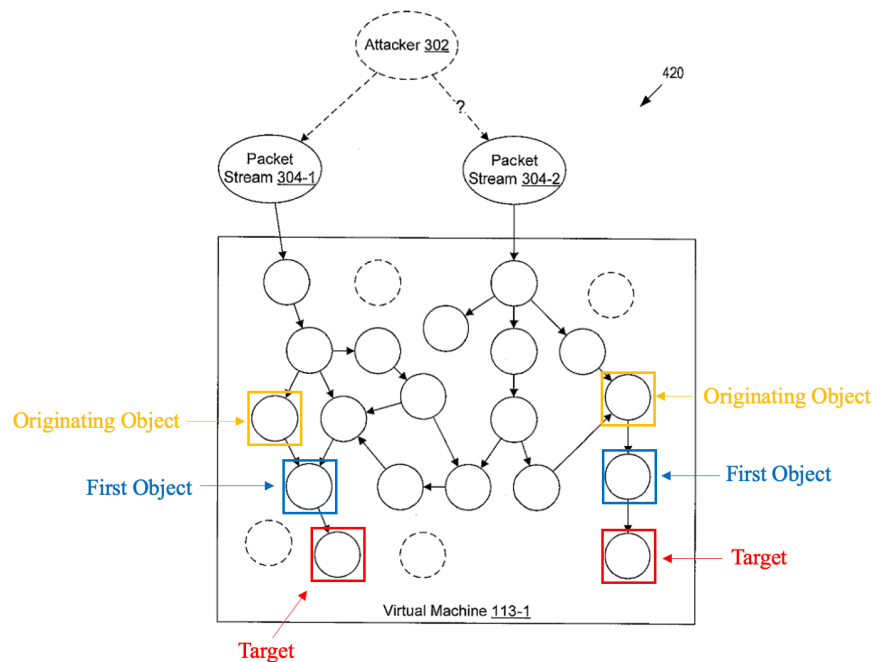


Figure 4E

211. Therefore, it is my opinion that *Capalik* discloses *heuristically displaying information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object.*

6. Claim 8

- (a) 8[P]. A system for providing forensic visibility into a system, comprising:

212. *Capalik* discloses “systems and methods for protecting computer networks” including “systems and methods for analyzing malicious activities on a computer system in order to better protect the computer system from future malicious activity.” *Capalik*, [0004], Abstract. This distributed computer system is depicted below in Figure 1:

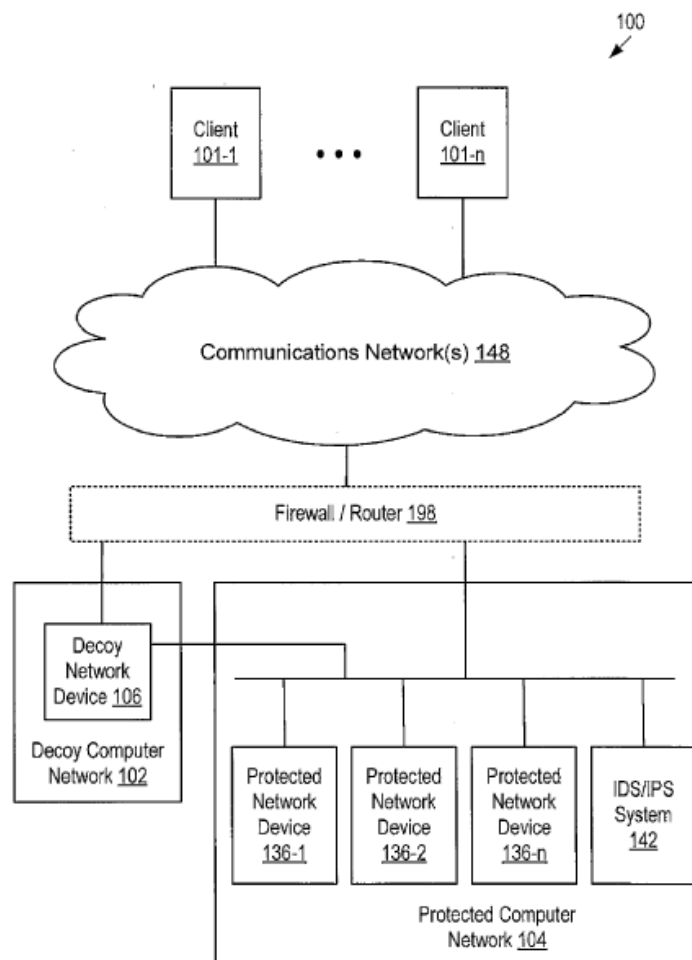


Figure 1

Id., Fig. 1. This system “generates and retrieves forensic data (or fingerprint)” relating to system events. *Id.*, [0048]; *see also id.*, [0063] (analyzing “raw forensic information” for a “forensic analysis”). *Capalik’s* system, therefore, provides forensic visibility into a system by forensically analyzing system events. *See also, id.*, [0039] (disclosing its decoy network device prepares the fingerprint event data “for display” such as “on a GUI of a client 101” thereby providing visibility into these events).

(b) [8(a)] a device including:

213. As illustrated above, this system includes a **decoy network device 106**. Fig. 1; *see also id.*, [0030]. This decoy network device and its associated components are illustrated below in Figure 2:

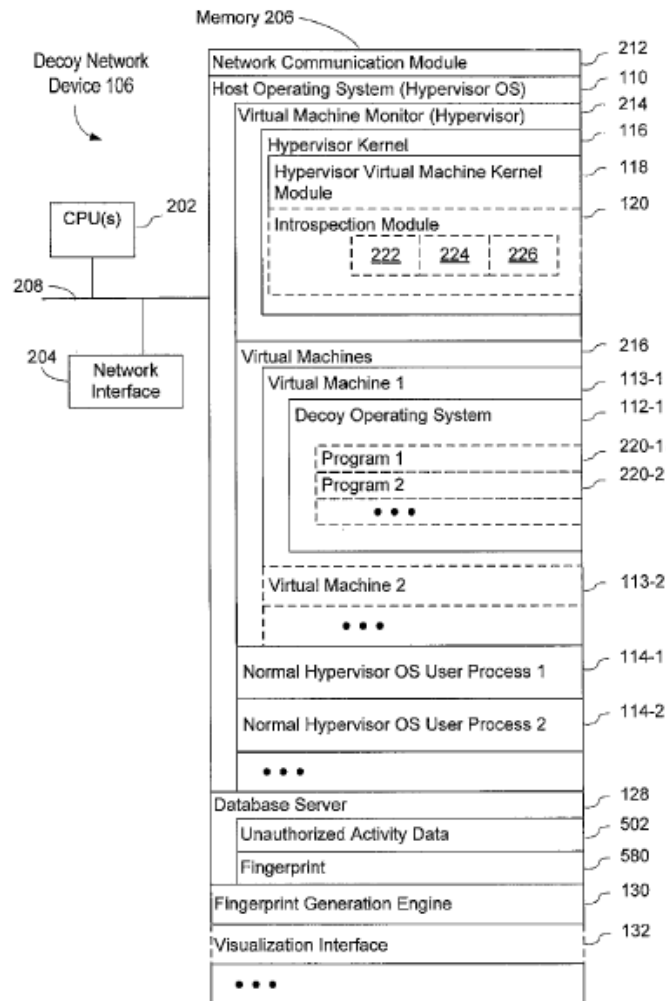


Figure 2

Id., Fig. 2.

(c) [8(b)] a communication interface;

214. Capalik's decoy network device includes "one or more network or other **communications interfaces 204**[" Capalik, [0033]. This communication interface is highlighted below:

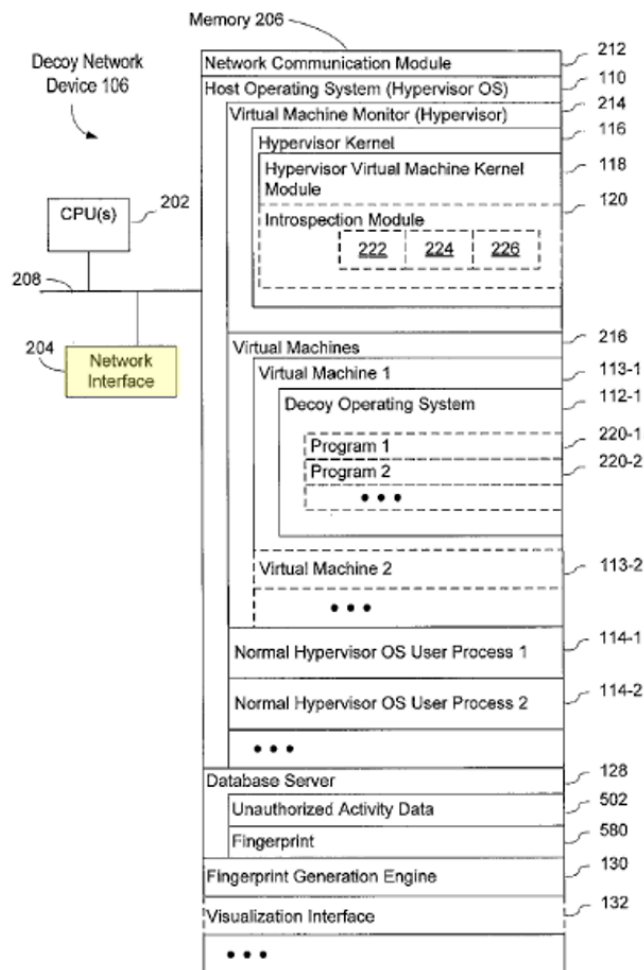


Figure 2

Id., Fig. 2.

(d) [8(c)] a processor;

215. *Capalik* discloses the decoy network device includes “one or more **processing units (CPUs)**[.]” *Capalik*, [0033]. CPU 202 is highlighted below in

Figure 2:

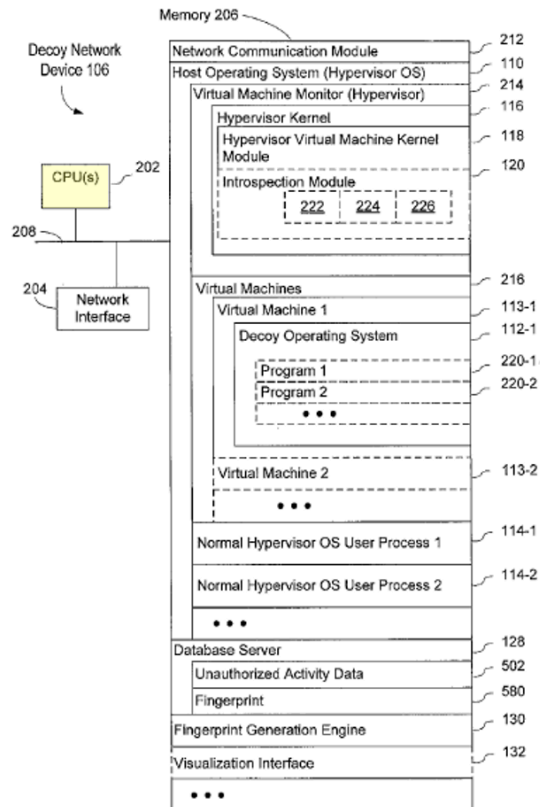


Figure 2

Id., Fig. 2.

(e) [8(d)] data storage; and

216. *Capalik* discloses the decoy network device includes “**memory 206**[.]” *Capalik*, [0033]. This memory may be DRAM, SRAM, DDR RAM, non-volatile memory, etc. *Id.*, [0034]. Memory 206 is shown below, boxed in red:

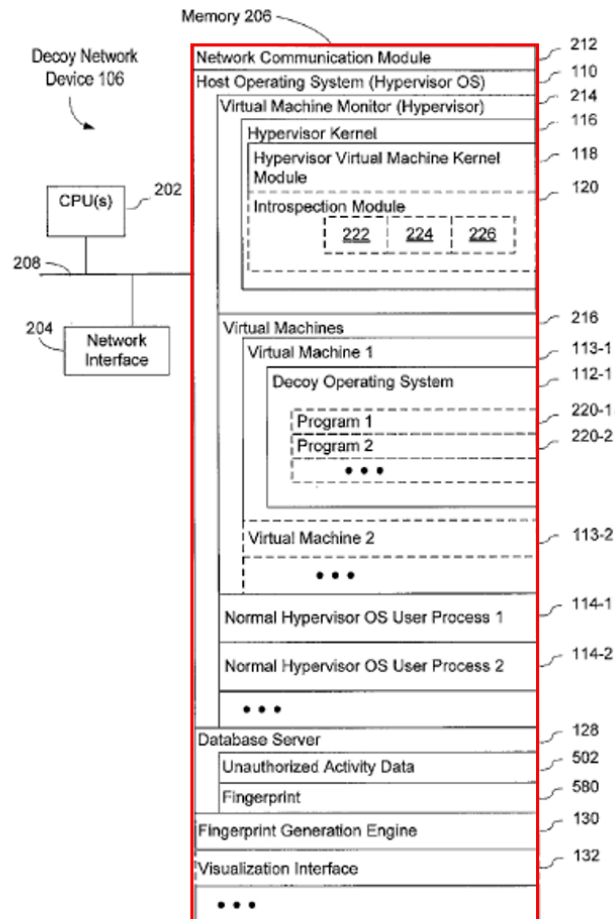


Figure 2

Id., Fig. 2.

- (f) [8(e)] a sensor agent stored on the data storage that is executable by the processor, wherein the sensor agent is operable to:

217. *Capalik* discloses “memory 206” stores a set of modules, including a network communications module 212, host operating system 110, database server 128, fingerprint generation engine 130, and visualization interface 132. *See generally Capalik*, [0034]-[0053]. These modules are shown above, stored in memory in Figure 2. These modules are collectively a *sensor agent* executable by

CPU 202 operable to perform the claimed functions. *Capalik* explains that its “modules...need not be implemented as separate software programs” such that they can be “combined or otherwise re-arranged.” *Capalik*, [0054]. A skilled artisan, therefore, would have understood these modules may collectively be implemented as a single sensor agent.

- (g) *[8(e)(i)-(v)]* gather one or more events defining an action of a first object acting on a target; generate a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object; obtain a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform Resource Locator (URL) information, wherein the global perspective for one or more related events to the at least one event across a network; assemble an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmit the assembled event line utilizing the communication interface.

218. I note that limitations 8(e)(i)-8(e)(v) are identical in substance to limitations [1(a)-1(e)] above. These limitations of claim 8 are obvious for the same reasons described in my opinions therein.

7. Claim 9 – The system of claim 8, wherein the at least one of the one or more events is executed by the processor.

219. As I discussed above, *Capalik* discloses that “memory 206” stores a set of modules, including a network communications module 212, host operating system 110, database server 128, fingerprint generation engine 130, and visualization interface 132. *See generally Capalik*, [0034]-[0053]. These modules are shown above, stored in memory in Figure 2. These modules are used to execute events and are executable by CPU 202 operable to perform the disclosed functions.

8. Claim 10 – The system of claim 8, further comprising: a global perspective information server; and an end recipient device, wherein the information associated with the one or more of the first object and the originating object is obtained from the global perspective information server and the sensor agent transmits the event line to the end recipient device.

220. As explained above, *Capalik* generally discusses the functions of claim 8’s system effectuated by the modules of the decoy network device 106. *Capalik* further discloses these functional elements, shown in Figure 2 above, can be “grouped or split among various components[.]” *Capalik*, [0055]. For example, *Capalik* contemplates splitting these modules into “distinct computing devices” such that one or more blocks of Figure 2 above can be **“implemented on one or more servers designed to provide the described functionality.”** *Id.*, [0056].

221. In this configuration, a skilled artisan would have understood the server could include functions of the introspection module to determine associations

between data components. In this way, this server constitutes a *global perspective information server* because it generates information associated with the IP Address of the device where Object 1 is executed and/or malware determination (type 516). Decoy network device 106 would then obtain this information from the server and include this information in the fingerprint for transmission by the sensor agent to an *end recipient device* (e.g., protected device, IDS/IPS) as discussed above in my opinions concerning claim 2. Moreover, this fingerprint can be sent to a user (e.g., client 101-1-101-n illustrated in Figure 1) for display on the user's GUI, which is another example of an *end recipient device*. *Capalik*, [0039].

9. *Claim 12 – The system of claim 8, wherein the at least one event includes information uniquely identifying the object performing the action, information uniquely identifying the action being performed, and information uniquely identifying the target upon which the act is being performed.*

222. As I explained above, *Capalik* discloses gathering event information, including “unauthorized activity entries 505,” comprising: “**an activity source 512**, which calls, accesses, or executes the activity target 514;” “**an activity target 514**, which is called, accessed, or executed by the activity source 512;” and “**an action 515**,” which represents an “operation occurring between the activity source 512 and the activity target 514.” *See above*, claim 4.

10. *Claim 15 – The system of claim 12, wherein the end recipient heuristically displays information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the*

act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.

223. As above, I have been informed that this limitation recites a disjunctive list of alternatives such that the prior art need only disclose one of the recited options to render this limitation unpatentable, where this limitation claims “the global perspective according to one or more of” “a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.” To render this limitation unpatentable, the prior art need only disclose one of the items in the above list.

224. As I explained above in my opinions concerning claim 7, *Capalik* discloses its methods further include “**displaying** at least a subset of the activities (e.g., **action sources**, **action targets**, and **actions**)” on a decoy network device’s visualization interface (*Capalik*, [0011]) including graphical representations showing associations between these activities—i.e., **the associations can be visually distinguished** (e.g., **a respective bold arrow representing a malignant association**...” *Id.*, [0074]. As above, a skilled artisan would have understood this includes displaying *information identifying the first object performing the action, information identifying the action being performed, information identifying the*

target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object.

225. *Capalik* also discloses a process for generating a fingerprint and transmitting this fingerprint to other computer systems on the network. *See above*, claim 1(d). For example, this fingerprint can optionally include the same information as above, illustrated in the exemplary fingerprint shown below:

580-1
↙

Status <u>506</u>				
source 1 <u>512-1</u>	target 1 <u>514-1</u>	Type <u>516-1</u>	Time Stamp <u>520-1</u>	505-1
source 2 <u>512-2</u>	target 2 <u>514-2</u>	Type <u>516-2</u>	Time Stamp <u>520-2</u>	505-2
source 3 <u>512-3</u>	target 3 <u>514-3</u>	Type <u>516-3</u>	Time Stamp <u>520-3</u>	505-3
source 4 <u>512-4</u>	target 4 <u>514-4</u>	Type <u>516-4</u>	Time Stamp <u>520-4</u>	505-4
source 5 <u>512-5</u>	target 5 <u>514-5</u>	Type <u>516-5</u>	Time Stamp <u>520-5</u>	505-5
⋮	⋮	⋮	⋮	

Figure 5B

Capalik, Figure 5B. Fingerprint 580 includes “an **activity source 512**, an **activity target 514**, an (optional) **association type 516**, and an (optional) time stamp 520.” *Id.*, [0090]. This fingerprint can also include an “**action 515**,” as described above. *See above*, claim 1(d).

226. *Capalik* explains this fingerprint can be sent to a user for subsequent display on the user's GUI. *Capalik*, [0039]. The user is boxed in red below in annotated Figure 1:

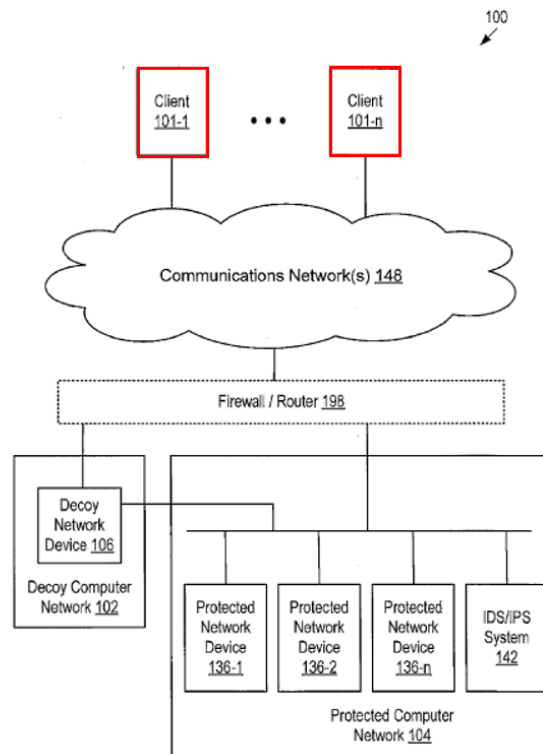


Figure 1

Id., Fig. 1; [0039] (describing preparing the fingerprint event data “for display” “on a GUI of a client 101”). A skilled artisan would have understood, therefore, that the same information discussed above displayed on the decoy network device’s visualization interface would similarly be displayed on a user’s GUI (i.e., heuristically displayed by an end recipient).

11. Claim 16

- (a) *16[P]-[16(e)] A system for providing forensic visibility into a system, comprising: a device including: a communication interface; a processor; data storage; and a sensor agent stored on the data storage that is executable by the processor, wherein the sensor agent is operable to:*

227. I note that limitations 16[P]-16(e) are identical in substance to limitations 8[P]-8(e) above. These limitations of claim 16 are obvious for the same reasons described in my opinions therein.

- (b) *[16(f)]-16(g)(i) gather one or more events defining an action of a first object acting on a target; generate a contextual state for at least one of the one or more events by correlating the at least one event to an originating object, the contextual state including an indication of the originating object of the first object and an indication of at least one of a device on which the first object is executed and a user associated with the first object;*

228. I note that limitations 16(f)-16(g)(i) are identical in substance to limitations 1(a)-1(b)(i) above. These system limitations of claim 16 are obvious for the same reasons described in my opinions therein.

- (c) *[16(h)] assemble an event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, the originating object, and global perspective information comprising information about one or more related events to the one or more events across a network; and*

229. Like I discussed in my opinions concerning claim 10, *Capalik* discloses implementations where the modules of decoy network device 106 is split into multiple components. *See above*, ¶¶220-221. Unlike the configuration of claim 10,

however, here, *Capalik*'s server does not implement components of the introspection module. Rather, the first component (e.g., a monitor device) does. Here, this first component would "collect[] unauthorized activity data" to send this unauthorized activity data to a server (*see Capalik*, [0056]) which includes the unauthorized activity data collected above in claim 1, including an **activity source 512**, an **activity target 514**, an (optional) **association type 516** (i.e., *global perspective information*) and **branching information** (i.e., originating object)). Thus, in this configuration, the first component sends an event line comprising the unauthorized activity data to a server.

(d) *[16(i)] transmit the assembled event line utilizing the communication interface.*

230. *See above, ¶196 (describing transmitting the event line to the server).*

A skilled artisan would have understood in this distributed configuration, the first component would send this event line to the server over the network (e.g., Internet) utilizing the communication interface.

12. *Claim 17 – The system of claim 16, further comprising a global perspective information server including: a processor; a communication interface; data storage; and a global perspective module, wherein the global perspective module is operable to: receive the event line from the device; obtain a global perspective for the at least one event by obtaining information associated with one or more of the first object and the originating object, the information including at least one of age, popularity, a determination as to whether the first object is malware, a determination as to whether the originating object is malware, Internet Protocol (IP) Address, and Uniform*

Resource Locator (URL) information; assemble a second event line including details associated with the at least one event, the details including information uniquely identifying each of the first object, the action of the first object, the target, and the originating object; and transmit the assembled second event line utilizing the communication interface to an end recipient.

231. As I explained in my opinions above in claim 16, *Capalik* discloses implementations where its modules of decoy network device 106 are split into multiple components, including a first component and a server (i.e., *global perspective information server*). See above, ¶229. One of skill in the art would have understood this server would include a *processor, communication interface, and data storage* in order to receive unauthorized activity data and generate a fingerprint. Indeed, it was common knowledge to those of skill in the art that a server generally comprises a CPU/processor, RAM, and a network communication subsystem. See, e.g., *Server Hardware, Software and Architecture* by Abeer El Hadi El Zein El Nahas, University of Khartoum, Electrical and Electronics Engineering Dept. (Feb. 2008) (Ex. 1025).

232. A skilled artisan would also have understood this server would include software enabling the server to receive the unauthorized activity data from the monitor device (i.e., *first event line*). This software constitutes a *global perspective module* as claimed. Upon receipt of this unauthorized activity data, this software on *Capalik's* server would obtain information from this first event line pertaining to the IP Address of the device where Object 1 is executed and/or malware determination

(type 516) (i.e., *global perspective*). The global perspective module would then perform the remaining functions disclosed in *Capalik*, including packaging this information in a *second event line* for transmission to an *end recipient*. For example, the server would send this information to a protected network device, an IDS/IPS, or a client 101 as described above. *See above*, ¶196.

13. *Claim 18 – The system of claim 17, wherein the end recipient heuristically displays information identifying the first object performing the action, information identifying the action being performed, information identifying the target upon which the act is being performed, and the global perspective according to one or more of a location associated with the first object; at least one characteristic of an outbound connection associated with the first object; a dropped executable file; an age of the first object; and a number of concurrent connections associated with the first object.*

233. *See above*, claim 15.

IV. CONCLUSION

234. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Date: November 29, 2022

By: 
Wenke Lee

Wenke Lee
Professor and John P. Imlay Jr. Chair
School of Cybersecurity and Privacy and School of Computer Science
College of Computing
Georgia Institute of Technology
Atlanta, Georgia 30332-0280
wenke@cc.gatech.edu
<http://www.cc.gatech.edu/~wenke/>

Contents

EDUCATIONAL BACKGROUND	2
EMPLOYMENT HISTORY	2
CURRENT FIELDS OF INTEREST	2
I. TEACHING	4
A. Courses Taught	4
B. Curriculum Development	6
C. Individual Student Guidance	8
II. RESEARCH AND CREATIVE SCHOLARSHIP	10
A. Theses	10
B. Published Journal Papers	10
C. Published Books and Parts of Books	11
D. Edited Proceedings and Books	11
E. Conference Presentations (refereed)	12
E.1 with Proceedings (refereed)	12
F. Workshop Presentations (refereed)	26
F.1 with Proceedings	26
G. Other	27
H. Research Proposals and Grants	29
I. Research Honors and Awards	36
III. SERVICE	38
A. Professional Activities	38
A.1 Membership and Activities in Professional Societies	38
A.2 Journal Edit-oral Board Activities	38
A.3 Conference Committee Activities	38
B. On-campus Georgia Tech Committees	42
IV. NATIONAL AND INTERNATIONAL PROFESSIONAL RECOGNITION	44
A. Patents	44
V. OTHER CONTRIBUTIONS	46
A. Seminar Presentations	46

EDUCATIONAL BACKGROUND

Degree	Year	University	Field
Ph.D.	1999	Columbia University	Computer Science
M.S.	1990	The City College of New York	Computer Science
B.S.	1988	Sun Yat-Sen University, China	Computer Science

EMPLOYMENT HISTORY

Title	Organization	Years
John P. Imlay Jr. Chair	College of Computing	3/2016–present
Director	Institute for Information Security & Privacy (IISP)	7/2015–7/2021
	Georgia Institute of Technology	
Director	Georgia Tech Information Security Center (GTISC)	8/2012–6/2015
	Georgia Institute of Technology	
Professor	College of Computing	2/2009–present
	Georgia Institute of Technology	
Associate Professor	College of Computing	2/2005–2/2009
	Georgia Institute of Technology	
Assistant Professor	College of Computing	8/2001–2/2005
	Georgia Institute of Technology	
Assistant Professor	Department of Computer Science	8/1999–8/2001
	North Carolina State University	
Research Assistant	Department of Computer Science	1994–1999
	Columbia University	
Research Staff Member	IBM T.J. Watson Research Center	5/1997–8/1997
Member of Technical Staff	AT&T Research	5/1996–8/1996
Senior Software Analyst	Intergraph Corporation	1991–1994
Software Analyst	Intergraph Corporation	1990–1991

CURRENT FIELDS OF INTEREST

Information Security

Main research thrusts are: “Malware analysis and detection” – developing “live” and “safe” environment to uncover malware behaviors and machine-learning based techniques to automatically generate detection models and forensic analysis rules; “Adversarial Machine Learning” – studying the vulnerabilities of machine learning based systems due to adversarial manipulations and developing countermeasures; “Software debloating and protocol subsetting” – developing program and protocol analysis and rewriting techniques to remove unnecessary features and code from program binaries; “Privacy-preserving biometric-based authentication and search” – researching efficient methods to protect the privacy of biometric data while enabling important security applications including re-

mote authentication and video surveillance; “Adaptive analysis framework for advanced persistent threats” – developing game-theoretic approaches to model and analyze advanced persistent threats.

I. TEACHING

A. Courses Taught

<u>Semester/Year</u>	<u>Course</u>	<u>Number of Students</u>	<u>Comments</u>
Fall 2021	OMS CS/Cybersecurity 6264 Information Security Labs: System and Network Defenses	25	Online
Fall 2021	OMS CS/Cybersecurity 6262 Network Security	360	Online
Fall 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	980	Online
Fall 2021	CS 6262 Network Security	40	At CoC
Summer 2021	OMS CS/Cybersecurity 6262 Network Security	200	Online
Summer 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	460	Online
Spring 2021	OMS CS/Cybersecurity 6264 Information Security Labs: System and Network Defenses	25	Online
Spring 2021	OMS CS/Cybersecurity 6262 Network Security	300	Online
Spring 2021	OMS CS/Cybersecurity 6035 Introduction to Information Security	800	Online
Fall 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	25	Online
Fall 2020	OMS CS/Cybersecurity 6262 Network Security	260	Online
Fall 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	850	Online
Fall 2020	CS 6262 Network Security	40	At CoC
Summer 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	20	Online
Summer 2020	OMS CS/Cybersecurity 6262 Network Security	150	Online
Summer 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	450	Online
Spring 2020	OMS CS/Cybersecurity 8803 Information Security Labs: System and Network Defenses	40	Online
Spring 2020	OMS CS/Cybersecurity 6262 Network Security	360	Online
Spring 2020	OMS CS/Cybersecurity 6035 Introduction to Information Security	884	Online
Fall 2019	OMS CS/Cybersecurity 6262 Network Security	156	Online
Fall 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	770	Online
Fall 2019	CS 6262 Network Security	43	At CoC
Summer 2019	OMS CS/Cybersecurity 6262 Network Security	129	Online
Summer 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	456	Online
Spring 2019	OMS CS/Cybersecurity 6262 Network Security	275	Online
Spring 2019	OMS CS/Cybersecurity 6035 Introduction to Information Security	938	Online

<u>Semester/Year</u>	<u>Course</u>	<u>Number of Students</u>	<u>Comments</u>
Fall 2018	OMSCS 6262 Network Security	154	Online
Fall 2018	OMSCS 6035 Introduction to Information Security	525	Online
Fall 2018	CS 6262 Network Security	47	At CoC
Summer 2018	OMSCS 6262 Network Security	191	Online
Summer 2018	OMSCS 6035 Introduction to Information Security	338	Online
Spring 2018	OMSCS 6262 Network Security	220	Online
Spring 2018	OMSCS 6035 Introduction to Information Security	428	Online
Fall 2017	OMSCS 6262 Network Security	300	Online
Fall 2017	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2017	CS 4235/6035 Introduction to Information Security	120	At CoC
Summer 2017	OMSCS 6262 Network Security	300	Online
Summer 2017	OMSCS 6035 Introduction to Information Security	350	Online
Spring 2017	OMSCS 6262 Network Security	300	Online
Spring 2017	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2016	OMSCS 6262 Network Security	300	Online
Fall 2016	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2016	CS 4235/6035 Introduction to Information Security	80	At CoC
Summer 2016	OMSCS 6035 Introduction to Information Security	350	Online
Spring 2016	OMSCS 6035 Introduction to Information Security	350	Online
Fall 2015	OMSCS 6035 Introduction to Information Security	230	Online
Fall 2015	CS 4235/6035 Introduction to Information Security	80	At CoC
Fall 2014	CS 4235/6035 Introduction to Information Security	80	At CoC
Spring 2013	CS 4235 Introduction to Information Security	50	At CoC
Fall 2012	CS 4235 Introduction to Information Security	41	At CoC
Spring 2012	CS 6262 Network Security	35	At CoC

<u>Semester/Year</u>	<u>Course</u>	<u>Number of Students</u>	<u>Comments</u>
Fall 2010	CS 3210 Operating System Design	25	At CoC
Spring 2010	CS 6262 Network Security	45	At CoC
Fall 2009	CS 4237 Computer and Network Security	15	At CoC
Spring 2009	CS 4237 Computer and Network Security	15	At CoC
Fall 2008	CS 6262 Network Security	55	At CoC
Spring 2008	CS 8803ANS Advanced Systems and Network Security	25	At CoC
Fall 2007	CS 4237 Computer and Network Security	20	At CoC
Spring 2007	CS 6262 Network Security	45	At CoC
Fall 2006	CS 6265 Information Security Lab (w/ Mustaque)	18	At CoC
Fall 2006	CS 4803 Computer and Network Security	30	At CoC
Spring 2006	CS 6262 Network Security	45	At CoC
Fall 2005	CS 4803 Computer and Network Security	30	At CoC
Spring 2005	CS 6262 Network Security	30	At CoC
Fall 2004	CS 4803K Computer and Network Security	30	At CoC
Spring 2004	CS 6262 Network Security	30	At CoC
Fall 2003	CS 4803K Computer and Network Security	35	At CoC
Spring 2003	CS 8803K Advanced Systems and Network Security	16	At CoC
Fall 2002	CS 6262 Network Security	28	At CoC
Spring 2002	CS 6262 Network Security	30	New at CoC
Fall 2000	CSC 591Q Special Topics: Network Security	65	At NC State
Fall 1999	CSC 591Q Special Topics: Network Security	45	At NC State
Summer 1998	CS 3139 Data Structures and Algorithms	20	At Columbia

B. Curriculum Development

CS 6264/OMS CS/Cybersecurity 6264: Information Security Labs: System and Network Defenses. This graduate-level course helps students develop both in-depth knowledge and hands-on skills in a number of important cybersecurity areas, including software security, malware, and threat analysis, end-point security, network security, web security, mobile security, and machine learning-based security analytics. The lecture materials of each topic area are drawn from the latest research papers and prototypes, and a comprehensive project is designed to help students master each area. The lecture materials explain the design principles of cutting-edge security tools, and the projects are designed to let students extend these tools. Most of the tools are in the open-source, and therefore, students can continue to build on and use these tools beyond this course. Recorded videos for on-line offering as part of OMSCS as well as OMS Cybersecurity.

CS 6262/OMS CS/Cybersecurity 6262: Network Security. Graduate-level course in network security with topics including large-scale attacks and impacts, penetration testing and security assessments, security of Internet protocols (IP, TCP, DNS, and BGP), advanced web security, advanced malware analysis, advanced network monitoring, Internet-scale threat analysis, bitcoins and cryptocurrencies, big data and security, cloud security, and attack-tolerant systems. Completely revamped

in 2016 with new course materials drawing from research papers and projects developed by my Ph.D. students. Recorded videos for on-line offering via Udacity as part of OMSCS as well as OMS Cybersecurity.

CS 4235/OMS CS/Cybersecurity 6035: Introduction to Information Security. Cross-listed undergraduate and graduate introductory course in information security. It teaches the basic concepts, principles, and fundamental approaches to secure computers and networks. Its main topics include: security basics, security management and risk assessment, software security, operating systems security, database security, cryptography algorithms and protocols, network authentication and secure network applications, malware, network threats and defenses, web security, mobile security, legal and ethical issues, and privacy. Completely revamped in 2015 with new materials covering the up-to-date threat models, attack methods, and new technologies and policy considerations. Recorded videos for on-line offering via Udacity as part of OMS CS as well as OMS Cybersecurity.

CS 6262: Network Security. Graduate-level course focusing on the fundamental principles as well as advanced techniques in network security. This course covers cryptography and its application to network and operating system security; authentication; security for electronic mail; Java security; Firewalls and intrusion detection.

CS 4237: Computer and Network Security. New undergraduate course (first offered as CS 4803 in Fall 2003) focusing on the fundamental principles and techniques in computer and network security.

CS 8803: Advanced Systems and Network Security. Graduate course on advanced topics such as network and host-based intrusion detection, botnet detection, software security, malware analysis, and virtual machine monitoring. Students are required to study research papers and work on research projects.

C. Individual Student Guidance

1. Postdoctoral Fellows Supervised

Erkam Uzun: 2021 -

Yisroel Mirsky: 2019 - 2021; Tenure-track Lecturer in the Department of Software and Information Systems Engineering, Ben-Gurion University of the Negev, Israel

Guangliang Yang: 2019 - 2021; Assistant Professor in the Department of Computer Science, Fudan University, Shanghai, China

Hong Hu: 2017-2019; Tenure-track Assistant Professor in the College of Information Sciences and Technology, Pennsylvania State University

Sangho Lee: 2015-2018; Researcher in Microsoft Research (MSR) Redmond

Tielei Wang: 2011 - 2014; Start-up in China

Simon Pak Ho Chung: 2011 -2014; Research Scientist in SCS at Georgia Tech

Roberto Perdisci: 2009 - 2010; Professor with tenure in the Department of Computer Science, University of Georgia

Daniel Xiapu Luo: 2008 - 2010; Associate Professor with tenure at Hong Kong Polytechnic University

2. Ph.D. Students Supervised

Xinzhou Qin: Graduated in Summer 2005; Juniper Networks

Yian Huang: Graduated in Summer 2006; Google

Prahlad Fogla: Graduated in Summer 2007; Google

Guofei Gu: Graduated in Summer 2008; Professor with tenure in the Department of Computer Science, Texas A&M University

Bryan Payne: Graduated in Summer 2010; Director of Product & Application Security, Netflix

Monirul Sharif: Graduated in Fall 2010; Engineering Manager III (L7), Google

Kapil Singh: Graduated in Summer 2011; Research Staff Member, IBM T.J. Watson Research Center

Manos Antonakakis: Graduated in Spring 2012; Associate Professor with tenure in ECE at Georgia Tech

Martim Carbone: Graduated in Summer 2012; Staff Engineer, VMware

Junjie Zhang: Graduated in Summer 2012; Associate Professor with tenure in the Department of Computer Science and Engineering, Wright State University

Long Lu: Graduated in Summer 2013; Associate Professor with tenure in the College of Computer and Information Science, Northeastern University

Brendan Dolan-Gavitt: Graduated in Summer 2014; tenure-track Assistant Professor in the Department of Computer Science and Engineering, NYU Tandon School of Engineering (Polytechnic Institute)

Yacin Nadji: Graduated in Summer 2015; Corelight

Xinyu Xing: Graduated in Summer 2015; Associate Professor of Computer Science, Northwestern University

Chengyu Song: Graduated in Summer 2016; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, UC Riverside

Byoungyoung Lee: Graduated in Summer 2016; Associate Professor in the Department of Electrical and Computer Engineering, Seoul National University, South Korea

Yeongjin Jang: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Electrical Engineering and Computer Science, Oregon State University

Kangjie Lu: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, University of Minnesota

Wei Meng: Graduated in Summer 2017; Tenure-track Assistant Professor in the Department of Computer Science and Engineering, The Chinese University of Hong Kong

Yizheng Chen: Graduated in Fall 2017; Post-doc in EECS, UC Berkeley

Yang Ji: Graduated in Fall 2019; Research Scientist in Palo Alto Networks

Ruian Duan: Graduated in Fall 2019; Research Scientist in Palo Alto Networks

Erkam Uzun: Graduated in Summer 2021; Post-doc at Georgia Tech

Chenxiong Qian: Graduated in Summer 2021; Tenure-track Assistant Professor in the Department of Computer Science, The University of Hong Kong

Evan Downing: In Progress

Kyuhong Park: In Progress

Carter Yagemann: In Progress

Joey Allen: In Progress

Matthew Landen: In Progress

Burak Sahin: In Progress

Yongheng Chen: In Progress

Feng Xiao: In Progress

Haoran Wang: In Progress

Ding Zhang: In Progress

Zhang Yang: In Progress

II. RESEARCH AND CREATIVE SCHOLARSHIP

A. Theses

Ph.D. Thesis

Title: A Data Mining Framework for Constructing Features and Models for Intrusion Detection Systems.

Date Completed: June 1999

Advisor: Professor Salvatore J. Stolfo.

University: Columbia University

B. Published Journal Papers (refereed)

1. Yisroel Mirsky and Wenke Lee. The Creation and Detection of Deepfakes: A Survey. *ACM Computing Surveys*. 54(1), 2021.
2. Kangjie Lu, Meng Xu, Chengyu Song, Taesoo Kim, and Wenke Lee. Stopping Memory Disclosures via Diversification and Replicated Execution. *IEEE Transactions on Dependable and Secure Computing (TDSC)*. 18(1). 2021.
3. Shana Moothedath, Dinuka Sahabandu, Joey Allen, Andrew Clark, Linda Bushnell, Wenke Lee, and Radha Poovendran. A Game-Theoretic Approach for Dynamic Information Flow Tracking to Detect Multistage Advanced Persistent Threats. *IEEE Transactions on Automatic Control*. 65(12): 5248-5263, 2020.
4. Junjie Zhang, Roberto Perdisci, Wenke Lee, Unum Sarfraz, and Xiapu Luo. Building a Scalable System for Stealthy P2P-Botnet Detection. *IEEE Transactions on Information Forensics and Security*, January 2014.
5. Roberto Perdisci, Davide Ariu, Prahlad Fogla, Giorgio Giacinto, and Wenke Lee. McPAD: A Multiple Classifier System for Accurate Payload-Based Anomaly Detection. *Computer Networks*, Vol. 53, 2009.
6. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. Classification of Packed Executables for Accurate Computer Virus Detection. *Pattern Recognition Letters* Vol. 29, No. 14 (October 2008).
7. Prahlad Fogla and Wenke Lee. q-Gram Matching Using Tree Models. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 18, No. 4 (April 2006).
8. W. Fan, M. Miller, S. Stolfo, Wenke Lee, and P. Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. *Knowledge and Information Systems*, Vol. 6, No. 5 (September 2004), Springer.
9. Yongguang Zhang, Wenke Lee, and Yian Huang. Intrusion Detection Techniques for Mobile Wireless Networks. *ACM/Kluwer Wireless Networks Journal (ACM WINET)*, 9(5), September 2003.

10. Joao B.D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, and Raman K. Mehra. Proactive Intrusion Detection and Distributed Denial of Service Attacks - A Case Study in Security Management. *Journal of Network and Systems Management*. 10(2), June 2002.
11. Wenke Lee, Wei Fan, Matt Miller, Sal Stolfo, and Erez Zadok. Toward Cost-Sensitive Modeling for Intrusion Detection and Response. *Journal of Computer Security*, 10(1,2), 2002.
12. Wenke Lee and Sal Stolfo. A Framework for Constructing Features and Models for Intrusion Detection Systems. *ACM Transactions on Information and System Security*, 3(4), November 2000.
13. Wenke Lee, Sal Stolfo, and Kui Mok. Adaptive Intrusion Detection: A Data Mining Approach. *Artificial Intelligence Review*, Kluwer Academic Publishers, 14(6):533-567, December 2000.
14. Wenke Lee and Gail E. Kaiser. Interfacing Oz with the PCTE OMS: A Case Study of Integrating a Legacy System with a Standard Object Management System. *Journal of Systems Integration*, Kluwer Academic Publishers, 9(4):329-358, December 1999.

C. Published Books and Parts of Books

1. Xinzhou Qin and Wenke Lee. Discovering Novel Attack Strategies from INFOSEC Alerts. *Data Warehousing and Data Mining Techniques for Cyber Security*. Anoop Singhal (eds), Springer, 2007.
2. Yongguang Zhang and Wenke Lee. Security in Mobile Ad-Hoc Networks. *Ad Hoc Networks: Technologies and Protocols*. P. Mohapatra and S. Krishnamurthy (eds), Springer, 2004.
3. Xinzhou Qin, Wenke Lee, Lundy Lewis, Joao B.D. Cabrera. Using MIB II Variables for Network Intrusion Detection. *Applications of Data Mining in Computer Security*. D. Barbara and S. Jajodia (eds), Kluwer Academic Publishers, May 2002.
4. Joao B.D. Cabrera, Lundy Lewis, Xinzhou Qin, Wenke Lee, Raman K. Mehra. Proactive Intrusion Detection - A Study on Temporal Data Mining. *Applications of Data Mining in Computer Security*. D. Barbara and S. Jajodia (eds), Kluwer Academic Publishers, May 2002.
5. Wenke Lee, Sal Stolfo, and Kui Mok. Algorithms for Mining System Audit Data. *Data Mining, Rough Sets, and Granular Computing*. T. Y. Lin, Y. Y. Yao, and L. A. Zadeh (eds), Physica-Verlag, 2002.
6. Wenke Lee and Naser Barghouti. Jadve: An Extensible Data Visualization Environment. In *Object-Oriented Applications Frameworks*, M. Fayad, D. Schmidt, and R. Johnson (eds), John Wiley & Sons, 1999.

D. Edited Proceedings and Books

1. *Botnet Detection: Countering the Largest Security Threat (Advances in Information Security)*, Wenke Lee, Cliff Wang, and David Dagon (Eds.), Springer, 2007.

2. *Proceedings of the 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001)*, Wenke Lee, Ludovic Me, and Andreas Wespi (Eds.), Lecture Notes in Computer Science, Vol. 2212, Springer, 2001.

E. Conference Presentations (refereed)

E.1. Conference Presentations with Proceedings

1. Carter Yagemann, Simon Chung, Brendan Saltaformaggio, and Wenke Lee. Automated Bug Hunting with Data-Driven Symbolic Root Cause Analysis. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2021.
2. Carter Yagemann, Mohammad Nouredine, Wajih Hassan, Simon Chung, Adam Bates, and Wenke Lee. Validating the Integrity of Audit Logs Against Execution Repartitioning Attacks. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2021.
3. Erkam Uzun, Simon P. Chung, Vladimir Kolesnikov, Alexandra Boldyreva, and Wenke Lee. Fuzzy Labeled Private Set Intersection with Applications to Private Real-Time Biometric Search. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.
4. Evan Downing, Kyuhong Park, Yisroel Mirsky, and Wenke Lee. DeepReflect: Discovering Malicious Functionality through Binary Reconstruction. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.
5. Carter Yagemann, Matthew Pruett, Simon P. Chung, Kennon Bittick, Brendan Saltaformaggio, and Wenke Lee. ARCUS: Symbolic Root Cause Analysis of Exploits in Production Systems. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.
6. Feng Xiao, Jianwei Huang, Yichang Xiong, Guangliang Yang, Hong Hu, Guofei Gu, and Wenke Lee. Abusing Hidden Properties to Attack the Node.js Ecosystem. In *Proceedings of the 2021 USENIX Security Symposium*. 2021.
7. Kyuhong Park, Burak Sahin, Yongheng Chen, Jisheng Zhao, Evan Downing, Hong Hu, and Wenke Lee. Identifying Behavior Dispatchers for Malware Analysis. In *Proceedings of the 16th ACM ASIA Conference on Computer and Communications Security (ACM AsiaCCS 2021)*.
8. Erkam Uzun, Carter Yagemann, Simon P. Chung, Vladimir Kolesnikov, and Wenke Lee. Cryptographic Key Derivation from Biometric Inferences for Remote Authentication. In *Proceedings of the 16th ACM ASIA Conference on Computer and Communications Security (ACM AsiaCCS 2021)*.
9. Dongsong Yu, Guangliang Yang, Guozhu Meng, Xiaorui Gong, Xiu Zhang, Xiaobo Xiang, Xiaoyu Wang, Yue Jiang, Kai Chen, Wei Zou, Wenke Lee, and Wenchang Shi. SEPAL: Towards a Large-scale Analysis of SEAndroid Policy Customization. In *Proceedings of The Web Conference 2021 (WWW 2021)*.

10. Yongheng Chen, Rui Zhong, Hong Hu, Hangfan Zhang, Yupeng Yang, Dinghao Wu, and Wenke Lee. One Engine to Fuzz 'em All: Generic Language Processor Testing with Semantic Validation. In *Proceedings of the 41st IEEE Symposium on Security and Privacy (Oakland)*. 2021.
11. Ruian Duan, Omar Alrawi, Ranjita Pai Kasturi, Ryan Elder, Brendan Saltaformaggio, and Wenke Lee. Towards Measuring Supply Chain Attacks on Package Managers for Interpreted Languages. In *Proceedings of the Network and Distributed System Security Symposium (NDSS)*. 2021.
12. Kangjie Lu, Meng Xu, Chengyu Song, Taesoo Kim, and Wenke Lee. Stopping Memory Disclosures via Diversification and Replicated Execution. *IEEE Transactions on Dependable and Secure Computing (TDSC)*. 18(1), 2021.
13. Carter Yagemann, Simon P. Chung, Erkam Uzun, Sai Ragam, Brendan Saltaformaggio, and Wenke Lee. On the Feasibility of Automating Stock Market Manipulation. In *Proceedings of the Annual Computer Security Applications Conference (ACSAC)*. 2020.
14. Chenxiong Qian, Hyungjoon Koo, ChangSeok Oh, Taesoo Kim, and Wenke Lee. Slimium: Debloating the Chromium Browser with Feature Subsetting. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.
15. Joey Allen, Zheng Yang, Matthew Landen, Raghav Bhat, Harsh Grover, Andrew Chang, Yang Ji, Roberto Perdisci, and Wenke Lee. Mnemosyne: An Effective and Efficient Postmortem Watering Hole Attack Investigation System. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.
16. Rui Zhong, Yongheng Chen, Hong Hu, Hangfan Zhang, Wenke Lee, and Dinghao Wu. SQUIRREL: Testing Database Management Systems with Language Validity and Coverage Feedback. In *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*. 2020.
17. Dinuka Sahabandu, Joey Allen, Shana Moothedath, Linda Bushnell, Wenke Lee, and Radha Poovendran. Quickest Detection of Advanced Persistent Threats: A Semi-Markov Game Approach. In *Proceedings of the ACM/IEEE International Conference on Cyber-Physical Systems (ICCPS)*. 2020.
18. D. Sahabandu, S. Moothedath, J. Allen, A. Clark, L. Bushnell, Wenke Lee, and R. Poovendran. Dynamic Information Flow Tracking Games for Simultaneous Detection of Multiple Attackers. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Nice, France, December 2019.
19. S. Misra, S. Moothedath, H. Hosseini, J. Allen, L. Bushnell, Wenke Lee, and R. Poovendran. Learning Equilibria in Stochastic Information Flow Tracking Games with Partial Knowledge. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, Nice, France, December 2019.

20. Dinuka Sahabandu, Shana Moothedath, Joey Allen, Linda Bushnell, Wenke Lee, and Radha Poovendran. Stochastic Dynamic Information Flow Tracking Game with Reinforcement Learning. In *Proceedings of the 2019 Conference on Decision and Game Theory for Security*, Stockholm, Sweden, October 2019.
21. Chenxiong Qian, Hong Hu, Mansour Alharthi, Pak Ho Chung, Taesoo Kim, and Wenke Lee. RAZOR: A Framework for Post-deployment Software Debloating. In *Proceedings of the 28th USENIX Security Symposium*. Santa Clara, CA, August 2019.
22. Dinuka Sahabandu, Shana Moothedath, Linda Bushnell, Radha Poovendran, Joey Allen, Wenke Lee, and Andrew Clark. A Game Theoretic Approach for Dynamic Information Flow Tracking with Conditional Branching. In *Proceedings of the 2019 American Control Conference (ACC)*. Philadelphia, PA, July 2019.
23. Ruian Duan, Ashish Bijlani, Yang Ji, Omar Alrawi, Yiyuan Xiong, Moses Ike, Brendan Saltaformaggio, and Wenke Lee. Automating Patching of Vulnerable Open-Source Software Versions in Application Binaries. In *Proceedings of the 2019 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2019.
24. Joey Allen, Matthew Landen, Sanya Chaba, Yang Ji, Simon Pak Ho Chung, and Wenke Lee. Improving Accuracy of Android Malware Detection with Lightweight Contextual Awareness. In *Proceedings of the 34th Annual Computer Security Applications Conference (ACSAC)*. December, 2018.
25. Dinuka Sahabandu, Baicen Xiao, Andrew Clark, Sangho Lee, Wenke Lee, and Radha Poovendran. DIFT Games: Dynamic Information Flow Tracking Games for Advanced Persistent Threats. In *Proceedings of The 57th IEEE Conference on Decision and Control (CDC)*. Miami Beach, FL, December 2018.
26. Shana Moothedath, Dinuka Sahabandu, Andrew Clark, Sangho Lee, Wenke Lee, and Radha Poovendran. Multi-Stage Dynamic Information Flow Tracking Game. In *Proceedings of The 9th Conference on Decision and Game Theory for Security (GameSec)*. Seattle, WA, October 2018.
27. Hong Hu, Chenxiong Qian, Carter Yagemann, Simon Pak Ho Chung, Bill Harris, Taesoo Kim, and Wenke Lee. Enforcing Unique Code Target Property for Control-Flow Integrity. In *Proceedings of The 25th ACM Conference on Computer and Communications Security (CCS 2018)*. Toronto, Canada, October 2018.
28. Andrea Possemato, Andrea Lanzi, Simon Pak Ho Chung, Wenke Lee, and Yanick Fratantonio. ClickShield: Are You Hiding Something? Towards Eradicating Clickjacking on Android. In *Proceedings of The 25th ACM Conference on Computer and Communications Security (CCS 2018)*. Toronto, Canada, October 2018.
29. Yang Ji, Sangho Lee, Mattia Fazzini, Joey Allen, Evan Downing, Taesoo Kim, Alessandro Orso, and Wenke Lee. Enabling Refinable Cross-Host Attack Investigation with Efficient Data Flow Tagging and Tracking. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, August 2018

30. Wei Meng, Chenxiong Qian, Shuang Hao, Kevin Borgolte, Giovanni Vigna, and Christopher Kruegel, and Wenke Lee. Rampart: Protecting Web Applications from CPU-Exhaustion Denial-of-Service Attacks. In *Proceedings of the 27th USENIX Security Symposium*. Baltimore, MD, August 2018
31. Erkam Uzun, Simon Pak Ho Chung, Irfan Essa, and Wenke Lee. rtCaptcha: A Real-Time CAPTCHA Based Liveness Detection System. In *Proceedings of The 2018 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2018.
32. Antonio Bianchi, Yanick Fratantonio, Aravind Machiry, Christopher Kruegel, Giovanni Vigna, Simon Pak Ho Chung, and Wenke Lee. Broken Fingers: On the Usage of the Fingerprint API in Android. In *Proceedings of the 2018 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2018.
33. Ruian Duan, Ashish Bijlani, Meng Xu, Taesoo Kim, and Wenke Lee. Identifying Open-Source License Violation and 1-day Security Risk at Large Scale. In *Proceedings of The 24th ACM Conference on Computer and Communications Security (CCS 2017)*. Dallas, Texas, October 2017.
34. Yang Ji, Sangho Lee, Evan Downing, Weiren Wang, Mattia Fazzini, Taesoo Kim, Alessandro Orso, and Wenke Lee. RAIN: Refinable Attack Investigation with On-demand Inter-Process Information Flow Tracking. In *Proceedings of The 24th ACM Conference on Computer and Communications Security (CCS 2017)*. Dallas, Texas, October 2017.
35. Ren Ding, Chenxiong Qian, Chengyu Song, Bill Harris, Taesoo Kim, and Wenke Lee. Efficient Protection of Path-Sensitive Control Security. In *Proceedings of the 26th USENIX Security Symposium*. Vancouver, BC, Canada, August 2017.
36. Meng Xu, Kangjie Lu, Taesoo Kim, and Wenke Lee. Bunshin: Compositing Security Mechanisms through Diversification. In *Proceedings of the 2017 USENIX Annual Technical Conference*. Santa Clara, CA, July 2017.
37. Yanick Fratantonio, Chenxiong Qian, Pak Chung, and Wenke Lee. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop. In *Proceedings of The 2017 IEEE Symposium on Security and Privacy*. San Jose, CA, May 2017 (**Distinguished Practical Paper Award**).
38. Kangjie Lu, Marie-Therese Walter, David Pfaff, Stefan Nuernberger, Wenke Lee, and Michael Backes. Unleashing Use-Before-Initialization Vulnerabilities in the Linux Kernel Using Targeted Stack Spraying. In *Proceedings of The 2017 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2017.
39. Le Guan, Jun Xu, Shuai Wang, Xinyu Xing, Lin Lin, Heqing Huang, Peng Liu, and Wenke Lee. From Physical to Cyber: Escalating Protection for Personalized Auto Insurance. In *Proceedings of The 14th ACM Conference on Embedded Networked Sensor Systems (SenSys 2016)*. Stanford, CA, November 2016.

40. Kangjie Lu, Chengyu Song, Taesoo Kim, and Wenke Lee. UniSan: Proactive Kernel Memory Initialization to Eliminate Data Leakages. In *Proceedings of The 23rd ACM Conference on Computer and Communications Security (CCS 2016)*. Vienna, Austria, October 2016.
41. Yizheng Chen, Panagiotis Kintis, Manos Antonakakis, Yacin Nadji, David Dagon, Wenke Lee, and Michael Farrell. Financial Lower Bounds of Online Advertising Abuse - A Four Year Case Study of the TDSS/TDL4 Botnet. In *Proceedings of The 13th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2016)*. Sebastian, Spain, July 2016.
42. Chengyu Song, Hyungon Moon, Monjur Alam, Insu Yun, Byoungyoung Lee, Taesoo Kim, and Wenke Lee, Yunheung Paek. HDFI: Hardware-Assisted Data-Flow Isolation. In *Proceedings of The 2016 IEEE Symposium on Security and Privacy*. San Jose, CA, May 2016.
43. Wei Meng, Byoungyoung Lee, Xinyu Xing, and Wenke Lee. TrackMeOrNot: Enable Flexible Control on Web Tracking. In *Proceedings of The 25th International World Wide Web Conference (WWW), April 2016*. San Diego, CA, February 2016.
44. Wei Meng, Ren Ding, Simon P. Chung, Steven Han, and Wenke Lee. The Price of Free: Privacy Leakage in Personalized Mobile In-Apps Ads. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2016.
45. Kangjie Lu, Wenke Lee, Stefan Nrnberger, and Michael Backes. How to Make ASLR Win the Clone Wars: Runtime Re-Randomization. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February 2016.
46. Chengyu Song, Byoungyoung Lee, Kangjie Lu, William Harris, Taesoo Kim and Wenke Lee. Enforcing Kernel Security Invariants with Data Flow Integrity. In *Proceedings of The 2016 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2016.
47. Meng Xu, Yeongjin Jang, Xinyu Xing, Taesoo Kim, and Wenke Lee. UCognito: Private Browsing without Tears. In *Proceedings of The 22nd ACM Conference on Computer and Communications Security (CCS)*. Denver, CO, October 2015.
48. Kangjie Lu, Chengyu Song, Byoungyoung Lee, Simon P. Chung, Taesoo Kim, and Wenke Lee. ASLR-Guard: Stopping Address Space Leakage for Code Reuse Attacks. In *Proceedings of The 22nd ACM Conference on Computer and Communications Security (CCS)*. Denver, CO, October 2015.
49. Byoungyoung Lee, Chengyu Song, Taesoo Kim, and Wenke Lee. Type Casting Verification: Stopping an Emerging Attack Vector. In *Proceedings of The 24th USENIX Security Symposium*. Washington, D.C., August 2015. (Awarded the **Internet Defense Prize** by Facebook and USENIX)
50. Xinyu Xing, Wei Meng, Byoungyoung Lee, Udi Weinsberg, Anmol Sheth, Roberto Perdisci, and Wenke Lee. Unraveling the Relationship Between Ad-Injecting Browser Extensions and Malvertising. In *Proceedings of The 24th International World Wide Web Conference (WWW)*. Florence, Italy, May 2015.

51. Chengyu Song, Chao Zhang, Tielei Wang, Wenke Lee, and David Melski. Exploiting and Protecting Dynamic Code Generation. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.
52. Byoungyoung Lee, Chengyu Song, Yeongjin Jang, Tielei Wang, Taesoo Kim, Long Lu, and Wenke Lee. Preventing Use-after-free with Dangling Pointers Nullification. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.
53. Kangjie Lu, Zhichun Li, Vasileios P. Kemerlis, Zhenyu Wu, Long Lu, Cong Zheng, Zhiyun Qian, Wenke Lee, and Guofei Jiang. Checking More and Alerting Less: Detecting Privacy Leakages via Enhanced Data-flow Analysis and Peer Voting. In *Proceedings of The 2015 Network and Distributed System Security Symposium (NDSS)*. San Diego, CA, February, 2015.
54. Wei Meng, Xinyu Xing, Anmol Sheth, Udi Weinsberg, and Wenke Lee. Your Online Interests Pwned! A Pollution Attack Against Targeted Advertising. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*. Scottsdale, AZ, November 2014.
55. Yeongjin Jang, Chengyu Song, Simon Chung, Tielei Wang, and Wenke Lee. All-y Attacks: Exploiting Accessibility in Operating Systems. In *Proceedings of the 21st ACM Conference on Computer and Communications Security (CCS)*. Scottsdale, AZ, November 2014.
56. Billy Lau, Pak Ho Chung, Chengyu Song, Yeongjin Jang, Wenke Lee, and Alexandra Boldyreva. Mimesis Aegis: A Mimicry Privacy Shield - A System's Approach to Data Privacy on Public Cloud. In *Proceedings of the 23rd USENIX Security Symposium*. San Diego, CA, August 2014.
57. Tielei Wang, Yeongjin Jang, Yizheng Chen, Pak Ho Chung, Billy Lau, and Wenke Lee. On the Feasibility of Large-Scale Infections of iOS Devices. In *Proceedings of the 23rd USENIX Security Symposium*. San Diego, CA, August 2014.
58. Byoungyoung Lee, Long Lu, Tielei Wang, Taesoo Kim, and Wenke Lee. From Zygote to Morula: Fortifying Weakened ASLR on Android. In *Proceedings of the 35th IEEE Symposium on Security and Privacy (Oakland)*, San Jose, CA, May 2014.
59. Xinyu Xing, Wei Meng, Dan Doozan, Nick Feamster, Wenke Lee, and Alex C. Snoeren. Exposing Inconsistent Web Search Results with Bobble. In *Proceedings of The 2014 Passive and Active Measurement Conference (PAM)*, Los Angeles, CA, March 2014.
60. Yeongjin Jang, Simon P. Chung, Bryan D. Payne, and Wenke Lee. Gyrus: A Framework for User-Intent Monitoring of Text-Based Networked Applications. In *Proceedings of the 21st Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2014.
61. Yacin Nadjji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Beheading Hydras: Performing Effective Botnet Takedowns. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, November 2013.

62. Brendan Dolan-Gavitt, Tim Leek, Josh Hodosh, and Wenke Lee. Tappan Zee (North) Bridge: Mining Memory Accesses for Introspection. In *Proceedings of the 20th ACM Conference on Computer and Communications Security (CCS)*, Berlin, Germany, November 2013.
63. Xinyu Xing, Wei Meng, Dan Doozan, Alex C. Snoeren, Nick Feamster, and Wenke Lee. Take this Personally: Pollution Attacks on Personalized Services. In *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., August 2013.
64. Tielei Wang, Kangjie Lu, Long Lu, Simon Chung, and Wenke Lee. Jekyll on iOS: When Benign Apps Become Evil. In *Proceedings of the 22nd USENIX Security Symposium*, Washington, D.C., August 2013.
65. Yacin Nadji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Connected Colors: Unveiling the Structure of Criminal Networks. In *Proceedings of the 16th International Symposium on Research in Attacks, Intrusions and Defenses (RAID)*, St. Lucia, October 2013 (to appear).
66. Junjie Zhang, Yinglian Xie, Fang Yu, David Soukal, and Wenke Lee. Intention and Origination: An Inside Look at Large-Scale Bot Queries. In *Proceedings of The 20th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2013.
67. Charles Lever, Manos Antonakakis, Bradley Reaves, Patrick Traynor and Wenke Lee. The Core of the Matter: Analyzing Malicious Traffic in Cellular Carriers. In *Proceedings of The 20th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2013.
68. Long Lu, Zhichun Li, Zhenyu Wu, Wenke Lee, and Guofei Jiang. CHEX: Statically Vetting Android Apps for Component Hijacking Vulnerabilities. In *Proceedings of the 19th ACM Conference on Computer and Communications Security (CCS)*, Raleigh, NC. October 2012.
69. Martim Carbone, Matthew Conover, Bruce Montague, and Wenke Lee. Secure and Robust Monitoring of Virtual Machines through Guest-Assisted Introspection. In *Proceedings of the 15th International Symposium on Research in Attacks, Intrusions, and Defenses (RAID)*. Amsterdam, The Netherlands. September, 2012.
70. Manos Antonakakis, Roberto Perdisci, Yacin Nadji, Nikolaos Vasiloglou, Saeed Abu-Nimeh, Wenke Lee, and David Dagon. From Throw-Away Traffic to Bots: Detecting the Rise of DGA-Based Malware. In *Proceedings of the 21st USENIX Security Symposium*. Bellevue, WA. August 2012.
71. Kapil Singh, Helen Wang, Alexander Moshchuk, Collin Jackson, and Wenke Lee. Practical End-to-End Web Content Integrity. In *Proceedings of the 21st International World Wide Web Conference (WWW)*, Lyon, France, April 2012.
72. Xiapu Luo, Peng Zhou, Junjie Zhang, Roberto Perdisci, Wenke Lee, and Rocky K.C. Chang. Exposing Invisible Timing-Based Traffic Watermarks with BACKLIT. In *Proceedings of The 27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, FL, December 2011.

73. Yacin Nadjji, Manos Antonakakis, Roberto Perdisci, and Wenke Lee. Understanding the Prevalence and Use of Alternative Plans in Malware with Network Games. In *Proceedings of The 27th Annual Computer Security Applications Conference (ACSAC 2011)*, Orlando, FL, December 2011.
74. Long Lu, Roberto Perdisci, and Wenke Lee. SURF: Detecting and Measuring Search Poisoning. In *Proceedings of The 18th ACM Conference on Computer and Communications Security (CCS)*. Chicago, IL, October 2011.
75. Manos Antonakakis, Roberto Perdisci, Wenke Lee, Nick Vasiloglou, and David Dagon. Detecting Malware Domains at the Upper DNS Hierarchy. In *Proceedings of The 20th USENIX Security Symposium*, San Francisco, CA August 2011.
76. Junjie Zhang, Roberto Perdisci, Wenke Lee, Unam Sarfraz, and Xiapu Luo. Detecting Stealthy P2P Botnets Using Statistical Traffic Fingerprints. In *Proceedings of The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011)*, Hong Kong, China, June 2011.
77. Xiapu Luo, Peng Zhou, Edmond W. W. Chan, Rocky K. C. Chang, and Wenke Lee. A Combinatorial Approach to Network Covert Communications with Applications in Web Leaks. In *Proceedings of The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011)*, Hong Kong, China, June 2011.
78. Brendan Dolan-Gavitt, Tim Leek, Michael Zhivich, Jon Giffin, and Wenke Lee. Virtuoso: Narrowing the Semantic Gap in Virtual Machine Introspection. In *Proceedings of The 2010 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2011.
79. Junjie Zhang, Jay Stokes, Christian Seifert, and Wenke Lee. ARROW: Generating Signatures to Detect Drive-By Downloads. In *Proceedings of The 20th International World Wide Web Conference (WWW)*, Hyderabad, India, March 2011.
80. Junjie Zhang, Xiapu Luo, Roberto Perdisci, Guofei Gu, Wenke Lee, and Nick Feamster. Boosting the Scalability of Botnet Detection Using Adaptive Traffic Sampling. In *Proceedings of The 6th ACM Symposium on Information, Computer and Communications Security (ASIACCS)*, Hong Kong, March 2011.
81. Xiapu Luo, Peng Zhou, Edmond W.W. Chan, Wenke Lee, Rocky K. C. Chang, and Roberto Perdisci. HTTPoS: Sealing Information Leaks with Browser-side Obfuscation of Encrypted Flows. In *Proceedings of The 18th Annual Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2011.
82. Qing Hui, Xiapu Luo, and Wenke Lee. Control of Low-Rate Denial-of-Service Attacks on Web Servers and TCP Flows. In *Proceedings of The 49th IEEE Conference on Decision and Control (CDC)*, Atlanta, GA, December 2010.
83. Long Lu, Vinod Yegneswaran, Phil Porras, and Wenke Lee. BLADE: An Attack-Agnostic Approach for Preventing Drive-By Malware Infections. In *Proceedings of The 17th ACM Conference on Computer and Communications Security (CCS)*, Chicago, IL, October 2010.

84. Xiapu Luo, Junjie Zhang, Roberto Perdisci, and Wenke Lee. On the Secrecy of Spread-Spectrum Flow Watermarks. In *Proceedings of The 15th European Symposium on Research in Computer Security (ESORICS)*, Athens, Greece, September 2010.
85. Manos Antonakakis, David Dagon, Xiapu Luo, Roberto Perdisci, and Wenke Lee. A Centralized Monitoring Infrastructure for Improving DNS Security. In *Proceedings of The 13th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Ottawa, Ontario, Canada, September 2010.
86. Manos Antonakakis, Roberto Perdisci, David Dagon, Wenke Lee, and Nick Feamster. Building a Dynamic Reputation System for DNS. In *Proceedings of The 19th USENIX Security Symposium*, Washington, DC, August 2010.
87. Kapil Singh, Samrit Sangal, Nehil Jain, Patrick Traynor, and Wenke Lee. Evaluating Bluetooth as a Medium for Botnet Command and Control. In *Proceedings of The 7th Conference on Detection of Intrusions and Malware Vulnerability Assessment (DIMVA)*, Bonn, Germany, July 2010.
88. Kapil Singh, Alexander Moshchuk, Helen J. Wang, and Wenke Lee. On the Incoherencies in Web Browser Access Control Policies. In *Proceedings of The 2010 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2010.
89. Roberto Perdisci, Wenke Lee, and Nick Feamster. Behavioral Clustering of HTTP-based Malware and Signature Generation using Malicious Network Traces. In *Proceedings of The 7th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, San Jose, CA, April 2010.
90. Roberto Perdisci, Igino Corona, David Dagon, and Wenke Lee. Detecting Malicious Flux Service Networks through Passive Analysis of Recursive DNS Traces. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.
91. Guofei Gu, Vinod Yegneswaran, Phillip Porras, Jennifer Stoll, and Wenke Lee. Active Botnet Probing to Identify Obscure Command and Control Channels. In *Proceedings of The 25th Annual Computer Security Applications Conference (ACSAC 2009)*, Honolulu, HI, December 2009.
92. Monirul Sharif, Wenke Lee, Weidong Cui, and Andrea Lanzi. Secure In-VM Monitoring Using Hardware Virtualization. In *Proceedings of The 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Chicago, IL, November, 2009.
93. Martim Carbone, Weidong Cui, Long Lu, Wenke Lee, Marcus Peinado, and Xuxian Jiang. Mapping Kernel Objects to Enable Systematic Integrity Checking. In *Proceedings of The 16th ACM Conference on Computer and Communications Security (CCS 2009)*, Chicago, IL, November, 2009.
94. Kapil Singh, Sumeer Bhola, and Wenke Lee. xBook: Redesigning Privacy Control in Social Networking Platforms. In *Proceedings of The 18th USENIX Security Symposium*, Montreal, Canada, August, 2009.

95. Roberto Perdisci, Manos Antonakakis, Xiapu Luo, and Wenke Lee. WSEC DNS: Protecting Recursive DNS Resolvers from Poisoning Attacks. In *Proceedings of The 39th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2009)*, Lisbon, Portugal, June 2009.
96. Monirul Sharif, Andrea Lanzi, Jon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators. In *Proceedings of The 2009 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2009. **(Best Student Paper Award)**
97. Andrea Lanzi, Monirul Sharif, and Wenke Lee. K-Tracer: A System for Extracting Kernel Malware Behavior. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.
98. David Dagon, Manos Antonakakis, Kevin Day, Xiapu Luo, Christopher P. Lee, and Wenke Lee. Recursive DNS Architectures and Vulnerability Implications. In *Proceedings of The 16th Annual Network and Distributed System Security Symposium (NDSS 2009)*, San Diego, CA, February 2009.
99. Roberto Perdisci, Andrea Lanzi, and Wenke Lee. McBoost: Boosting Scalability in Malware Collection and Analysis Using Statistical Classification of Executables. In *Proceedings of The 24th Annual Computer Security Applications Conference (ACSAC 2008)*, Anaheim, CA, December 2008.
100. Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: Malware Analysis via Hardware Virtualization Extensions. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.
101. David Dagon, Manos Antonakakis, Paul Vixie, Tatuya Jinmei, and Wenke Lee. Increased DNS Forgery Resistance Through 0x20-Bit Encoding. In *Proceedings of the 15th ACM Conference on Computer and Communications Security (CCS 2008)*, Alexandria, VA, October 2008.
102. Monirul Sharif, Vinod Yegneswaran, Hassen Saidi, Phillip Porras, and Wenke Lee. Eureka: A Framework for Enabling Static Malware Analysis. In *Proceedings of the 13th European Symposium on Research in Computer Security (ESORICS)*, Malaga, Spain, October 2008.
103. Guofei Gu, Roberto Perdisci, Junjie Zhang, and Wenke Lee. BotMiner: Clustering Analysis of Network Traffic for Protocol- and Structure-Independent Botnet Detection. In *Proceedings of The 17th USENIX Security Symposium (Security'08)*, San Jose, CA, July 2008.
104. Kapil Singh, Abhinav Srivastava, Jon Giffin, and Wenke Lee. Evaluating Email's Feasibility for Botnet Command and Control. In *Proceedings of the 38th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2008)*, Anchorage, Alaska, June 2008.
105. Bryan D. Payne, Martim Carbone, Monirul Sharif, and Wenke Lee. Lares: An Architecture for Secure Active Monitoring Using Virtualization. In *Proceedings of the 2008 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2008.

106. Guofei Gu, Alvaro A. Cardenas, and Wenke Lee. Principled Reasoning and Practical Applications of Alert Fusion in Intrusion Detection Systems. In *Proceedings of the ACM Symposium on InformAction, Computer and Communications Security (ASIACCS'08)*, Tokyo, Japan, March 2008.
107. David Dagon, Niels Provos, Chris Lee, and Wenke Lee. Corrupted DNS Resolution Paths: The Rise of a Malicious Resolution Authority. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.
108. Guofei Gu, Junjie Zhang, and Wenke Lee. BotSniffer: Detecting Botnet Command and Control Channels in Network Traffic. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.
109. Monirul Sharif, Andrea Lanzi, Jonathon Giffin, and Wenke Lee. Impeding Malware Analysis using Conditional Code Obfuscation. In *Proceedings of The 15th Annual Network and Distributed System Security Symposium (NDSS 2008)*, San Diego, CA, February 2008.
110. Bryan D. Payne, Martim Carbone, and Wenke Lee. Secure and Flexible Monitoring of Virtual Machines. In *Proceedings of The 23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, December 2007.
111. David Dagon, Guofei Gu, Chris Lee and Wenke Lee. A Taxonomy of Botnet Structures. In *Proceedings of The 23rd Annual Computer Security Applications Conference (ACSAC 2007)*, Miami Beach, FL, December 2007.
112. Guofei Gu, Zesheng Chen, Phillip Porras and Wenke Lee. Misleading and Defeating Importance-Scanning Malware Propagation. In *Proceedings of The 3rd International Conference on Security and Privacy in Communication Networks (SecureComm'07)*, Nice, France, September 2007.
113. Takehiro Takahashi and Wenke Lee. An Assessment of VoIP Covert Channel Threats. In *Proceedings of The 3rd International Conference on Security and Privacy in Communication Networks (SecureComm'07)*, Nice, France, September 2007.
114. Monirul Sharif, Kapil Singh, Jonathon Giffin and Wenke Lee. Understanding Precision in Host Based Intrusion Detection: Formal Analysis and Practical Models. In *Proceedings of The 10th International Symposium on Recent Advances in Intrusion Detection (RAID)*, Surfers Paradise, Australia, September 2007.
115. Guofei Gu, Phillip Porras, Vinod Yegneswaran, Martin Fong, Wenke Lee. BotHunter: Detecting Malware Infection Through IDS-Driven Dialog Correlation. In *Proceedings of The 16th USENIX Security Symposium (Security'07)*, Boston, MA, August 2007.
116. David Cash, Yan Zong Ding, Yevgeniy Dodis, Wenke Lee, Richard Lipton, and Shabsi Wal-fish. Intrusion-Resilient Key Exchange in the Bounded Retrieval Model. In *Proceedings of The Fourth IACR Theory of Cryptography Conference (TCC 2007)*, Amsterdam, The Netherlands, February 2007.

117. Roberto Perdisci, Guofei Gu, and Wenke Lee. Using an Ensemble of One-Class SVM Classifiers to Harden Payload-based Anomaly Detection Systems. In *Proceedings of The 2006 IEEE International Conference on Data Mining (ICDM '06)*, Hong Kong, China, December 2006.
118. Paul Royal, Mitch Halpin, David Dagon, Robert Edmonds, and Wenke Lee. PolyUnpack: Automating the Hidden-Code Extraction of Unpack-Executing Malware. In *Proceedings of The 22th Annual Computer Security Applications Conference (ACSAC 2006)*, Miami Beach, FL, December 2006.
119. Prahlad Fogla and Wenke Lee. Evading Network Anomaly Detection Systems: Formal Reasoning and Practical Techniques. In *Proceedings of The 13th ACM Conference on Computer and Communications Security (CCS 2006)*, Alexandria, VA, October 2006.
120. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Towards an Information-Theoretic Framework for Analyzing Intrusion Detection Systems. In *Proceedings of The 11th European Symposium Research Computer Security (ESORICS 2006)*, Hamburg, Germany, September 2006.
121. Prahlad Fogla, Monirul Sharif, Roberto Perdisci, Oleg Kolesnikov, and Wenke Lee. Polymorphic Blending Attacks. In *Proceedings of The 15th USENIX Security Symposium (SECURITY '06)*, Vancouver, B.C., Canada, August 2006.
122. Collin Mulliner, Giovanni Vigna, David Dagon, and Wenke Lee. Using Labeling to Prevent Cross-Service Attacks Against Smart Phones. In *Proceedings of The 3rd Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA 2006)*, Berlin, Germany, July 2006.
123. Hongmei Deng, Roger Xu, Jason H. Li, Frank Zhang, Renato Levy, and Wenke Lee. Agent-Based Cooperative Anomaly Detection for Wireless Ad Hoc Networks. In *Proceedings of The 12th International Conference on Parallel and Distributed Systems (ICPADS 2006)*, Minneapolis, Minnesota, July 2006.
124. Guofei Gu, Prahlad Fogla, Wenke Lee, and Douglas Blough. DSO: Dependable Signing Overlay. In *Proceedings of The 4th International Conference on Applied Cryptography and Network Security (ACNS '06)*, Singapore, June 2006.
125. Roberto Perdisci, David Dagon, Wenke Lee, Prahlad Fogla, and Monirul Sharif. Misleading Worm Signature Generators Using Deliberate Noise Injection (full paper). In *Proceedings of the 2006 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2006.
126. Guofei Gu, Prahlad Fogla, David Dagon, Wenke Lee, and Boris Skoric. Measuring Intrusion Detection Capability: An Information-Theoretic Approach. In *Proceedings of ACM Symposium on InformAction, Computer and Communications Security (ASIACCS '06)*, Taipei, Taiwan, March 2006.
127. David Dagon, Cliff Zou, and Wenke Lee. Modeling Botnet Propagation Using Time Zones. In *Proceedings of The 13th Annual Network and Distributed System Security Symposium (NDSS 2006)*, San Diego, CA, February 2006.

128. Yongguang Zhang, Yi-an Huang, and Wenke Lee. An Extensible Environment for Evaluating Secure MANET. In *Proceedings of The 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005)*, Athens, Greece, September 2005.
129. Jonathon T. Giffin, David Dagon, Somesh Jha, Wenke Lee, and Barton P. Miller. Environment-Sensitive Intrusion Detection. In *Proceedings of The 8th International Symposium on Recent Advances in Intrusion Detection (RAID 2005)*, Seattle, WA, September 2005.
130. David Dagon, Wenke Lee, and Richard Lipton. Protecting Secret Data from Insider Attacks. *Proceedings of The Ninth International Conference on Financial Cryptography and Data Security (FC'05)*, Roseau, Dominica, February, 2005.
131. Guofei Gu, David Dagon, Xinzhou Qin, Monirul I. Sharif, Wenke Lee, and George F. Riley. Worm Detection, Early Warning, and Response Based on Local Victim Information. *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, Tucson, Arizona, December 2004.
132. Xinzhou Qin and Wenke Lee. Attack Plan Recognition and Prediction Using Causal Networks. *Proceedings of the 20th Annual Computer Security Applications Conference (ACSAC 2004)*, Tucson, Arizona, December 2004.
133. Joao B.D. Cabrera, Jaykumar Gosar, Wenke Lee, and Raman K. Mehra. On the Statistical Distribution of Processing Times in Network Intrusion Detection. *Proceedings of the 43rd IEEE Conference on Decision and Control (CDC 2004)*, Bahamas, December 2004.
134. George F. Riley, Monirul I. Sharif, and Wenke Lee. Simulating Internet Worms. *Proceedings of the 12th Annual Meeting of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*, Volendam, The Netherlands, October 2004.
135. Yian Huang and Wenke Lee. Attack Analysis and Detection for Ad Hoc Routing Protocols. *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, Sophia Antipolis, France, September 2004.
136. David Dagon, Xinzhou Qin, Guofei Gu, Wenke Lee, Julian Grizzard, John Levin, and Henry Owen. HoneyStat: Local Worm Detection Using Honey pots. *Proceedings of the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004)*, Sophia Antipolis, France, September 2004.
137. Xinzhou Qin and Wenke Lee. Discovering Novel Attack Strategies from INFOSEC Alerts. *Proceedings of the 9th European Symposium on Research in Computer Security (ESORICS 2004)*, Sophia Antipolis, France, September 2004.
138. H. Feng, Jonathon T. Giffin, Yong Huang, Somesh Jha, Wenke Lee, and Barton P. Miller. Formalizing Sensitivity in Static Analysis for Intrusion Detection. *Proceedings of the 2004 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2004.

139. Xinzhou Qin and Wenke Lee. Statistical Causality Analysis of INFOSEC Alert Data. In *Proceedings of the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003)*, Pittsburgh, PA, September 2003.
140. H. Feng, Oleg Kolesnikov, Prahlad Fogla, Wenke Lee, and Weibo Gong. Anomaly Detection Using Call Stack Information. In *Proceedings of the 2003 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2003.
141. Yi-an Huang, Wei Fan, Wenke Lee, and Philip S. Yu. Cross-Feature Analysis for Detecting Ad-Hoc Routing Anomalies. In *Proceedings of the 23rd International Conference on Distributed Computing Systems (ICDCS)*, Providence, RI, May 2003.
142. Joao B.D. Cabrera, Wenke Lee, R.K. Prasanth, Lundy Lewis, and Raman K. Mehra. Optimization and Control Problems in Real Time Intrusion Detection. *Proceedings of the 41st IEEE Conference on Decision and Control (CDC 2002)*, Las Vegas, December, 2002.
143. Wenke Lee, Joao B.D. Cabrera, Ashley Thomas, Niranjana Balwalli, Sunmeet Saluja, and Yi Zhang. Performance Adaptation in Real-Time Intrusion Detection Systems. In *Proceedings of the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002)*, Zurich, Switzerland, October 2002.
144. Xinzhou Qin, Wenke Lee, Lundy Lewis, and Joao B.D. Cabrera. Integrating Intrusion Detection and Network Management. *Proceedings of the IEEE/IFIP Network Operations and Management Symposium (NOMS 2002)*, Florence, Italy, May 2002.
145. Wei Fan, Matt Miller, Sal Stolfo, Wenke Lee, and Phil Chan. Using Artificial Anomalies to Detect Unknown and Known Network Intrusions. *Proceedings of The First IEEE International Conference on Data Mining (ICDM)*, San Jose, CA, November 2001.
146. Wenke Lee, Sal Stolfo, Phil Chan, Eleazar Eskin, Wei Fan, Matt Miller, Shlomo HersHKop, and Junxin Zhang. Real Time Data Mining-based Intrusion Detection. *Proceedings of the 2001 DARPA Information Survivability Conference and Exposition (DISCEX II)*, Anaheim, CA, June 2001.
147. Wenke Lee and Dong Xiang. Information-Theoretic Measures for Anomaly Detection. *Proceedings of The 2001 IEEE Symposium on Security and Privacy*, Oakland, CA, May 2001.
148. Joao B.D. Cabrera, L. Lewis, X. Qin, Wenke Lee, Ravi Prasanth, B. Ravichandran, and Raman Mehra. Proactive Detection of Distributed Denial of Service Attacks Using MIB Traffic Variables - A Feasibility Study. *Proceedings of The Seventh IFIP/IEEE International Symposium on Integrated Network Management (IM 2001)*, Seattle, WA, May 2001.
149. Yongguang Zhang and Wenke Lee. Intrusion Detection in Wireless Ad-Hoc Networks. *Proceedings of The Sixth International Conference on Mobile Computing and Networking (MobiCom 2000)*, Boston, MA, August 2000.
150. Wei Fan, Wenke Lee, Sal Stolfo, and Matt Miller. A Multiple Model Cost-Sensitive Approach for Intrusion Detection. *Proceedings of The Eleventh European Conference on Machine Learning (ECML 2000)*, LNAI 1810, Barcelona, Spain, May 2000.

151. Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Phil Chan. Cost-based Modeling for Fraud and Intrusion Detection: Results from the JAM Project. *Proceedings of the 2000 DARPA Information Survivability Conference and Exposition (DISCEX '00)*, Hilton Head, SC, January 2000.
152. Wenke Lee, Sal Stolfo, and Kui Mok. Mining in a Data-flow Environment: Experience in Network Intrusion Detection. *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99)*, San Diego, CA, August 1999.
153. Wenke Lee, Sal Stolfo, and Kui Mok. A Data Mining Framework for Building Intrusion Detection Models. *Proceedings of the 1999 IEEE Symposium on Security and Privacy*, Oakland, CA, May 1999.
154. Wenke Lee, Sal Stolfo, and Kui Mok. Mining Audit Data to Build Intrusion Detection Models. *Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD '98)*, New York, NY, August 1998.
155. Wenke Lee and Sal Stolfo. Data Mining Approaches for Intrusion Detection. *Proceedings of the Seventh USENIX Security Symposium (SECURITY '98)*, San Antonio, TX, January 1998.
156. Sal Stolfo, Andreas Prodromidis, Shelley Tselepis, Wenke Lee, Wei Fan, and Phil Chan. JAM: Java Agents for Meta-learning over Distributed Databases. *Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD '97)*, Newport Beach, CA, August 1997.
157. Naser S. Barghouti, John Mocenigo, and Wenke Lee. Grappa: A GRAPH PACKage in Java. *Proceedings of the Fifth Annual Symposium on Graph Drawing (Graph Drawing '97)*, Rome, Italy, September 1997.
158. Wenke Lee, Gail Kaiser, Paul Clayton, and Eric Sherman. OzCare: A Workflow Automation System for Care Plans. *Proceedings of the American Medical Informatics Association Annual Fall Symposium*, Washington DC, October 1996.

F. Workshop Presentations (refereed)

F.1. Workshop Presentations with Proceedings

1. Yi-an Huang and Wenke Lee. Hotspot-Based Traceback for Mobile Ad Hoc Networks. In *Proceedings of The ACM Workshop on Wireless Security (WiSe 2005)*, Cologne, Germany, September 2005.
2. Monirul Sharif, George Riley, and Wenke Lee. Comparative Study between Analytical Models and Packet-Level Worm Simulations. In *Proceedings of The 19th Workshop on Parallel and Distributed Simulation (PADS 2005)*, Monterey, CA, June 2005.
3. Chris Clark, Wenke Lee, David Schimmel, Didier Contis, Mohamed Kone, and Ashley Thomas. A Hardware Platform for Network Intrusion Detection and Prevention. *The 3rd Workshop on Network Processors and Applications (NP3)*, Madrid, Spain, February 2004.

4. Yian Huang and Wenke Lee. A Cooperative Intrusion Detection System for Ad Hoc Networks. *ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN '03)*, Fairfax VA, October 2003.
5. Mustaque Ahamad, Wenke Lee, Ling Liu, Leo Mark, Edward Omicienski, Calton Pu, and Andre dos Santos. Guarding the Next Internet Frontier: Countering Denial of Information Attacks. *Proceedings of the 2002 New Security Paradigms Workshop*, Virginia Beach, Virginia, September 2002.
6. Xinzhou Qin, Wenke Lee, Lundy Lewis, and Joao B.D. Cabrera. Using MIB III Variables for Network Anomaly Detection - A Feasibility Study. *ACM Workshop on Data Mining for Security Applications*, Philadelphia, PA, November 2001.
7. Yongguang Zhang, Harrick Vin, Lorenzo Alvisi, Wenke Lee, and Son K. Dao. Heterogeneous Networking: A New Survivability Paradigm. *Proceedings of the 2001 New Security Paradigms Workshop*, Cloudcroft, New Mexico, September 2001.
8. Wenke Lee, Wei Fan, Matt Miller, Sal Stolfo, and Erez Zadok. Toward Cost-Sensitive Modeling for Intrusion Detection and Response. *ACM Workshop on Intrusion Detection Systems*, Athens, Greece, November 2000.
9. Wenke Lee, Rahul Nimbalkar, Kam Yee, Sunil Patil, Pragnesh Desai, Thuan Tran, and Sal Stolfo. A Data Mining and CIDF Based Approach for Detecting Novel and Distributed Intrusions. *Proceedings of The Third International Workshop on Recent Advances in Intrusion Detection (RAID 2000)*, LNCS 1907, Toulouse, France, October 2000.
10. Wenke Lee, Chris Park, and Sal Stolfo. Towards Automatic Intrusion Detection using NFR. *1st USENIX Workshop on Intrusion Detection and Network Monitoring*, April 1999.
11. Wenke Lee, Sal Stolfo, and Phil Chan. Learning Patterns from Unix Process Execution Traces for Intrusion Detection. *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, July 1997.
12. Sal Stolfo, Wei Fan, Wenke Lee, Andreas Prodromidis, and Phil Chan. Credit Card Fraud Detection Using Meta-Learning: Issues and Initial Results. *AAAI Workshop: AI Approaches to Fraud Detection and Risk Management*, July 1997.
13. Gail Kaiser and Wenke Lee. Pay No Attention to the Man Behind the Curtain. *NSF Workshop on Workflow and Process Automation*, May 1996.
14. Wenke Lee. Data Modeling and Management for Large Spatial Databases. *The Third International Workshop in Geographic Information Systems*, Beijing, China, August 1993.

G. Other

1. Matt Blaze, Sampath Kannan, Insup Lee, Oleg Sokolsky, Jonathan Smith, Angelos Keromytis, and Wenke Lee. Dynamic Trust Management. *IEEE Computer*, February 2009.

2. Martim Carbone, Diego Zamboni, and Wenke Lee. Taming Virtualization. *IEEE Security & Privacy*, vol. 6, no. 1, January/February 2008.
3. Bryan D. Payne, Reiner Sailer, Ramon Caceres, Ronald Perez, and Wenke Lee. A Layered Approach to Simplified Access Control in Virtualized Systems. *ACM SIGOPS Operating Systems Review*, 4(2), July 2007.
4. Wenke Lee. 2002. Applying Data Mining to Intrusion Detection: The Quest for Automation, Efficiency, and Credibility. *SIGKDD Explorations*, 4(2), December 2002.
5. Wenke Lee and Wei Fan. 2001. Mining System Audit Data: Opportunities and Challenges. *SIGMOD Record*, 30(4), December 2001.
6. Salvatore J. Stolfo, Wenke Lee, Philip K. Chan, Wei Fan, and Eleazar Eskin. 2001. Data Mining-Based Intrusion Detectors: An Overview of the Columbia IDS Project. *SIGMOD Record*, 30(4), December 2001.

H. Research Proposals and Grants

a. Approved and Funded

1. **Proof-driven Refinement of Infrastructure Software Modules (PRISM)**
Brendan Saltaformaggio (PI); co-PIs: Wenke Lee, Taesoo Kim, Alessandro Orso, Qirun Zhang, Yisroel Mirsky, Martin Osterloh, Jason Li, Michael Brown, Sukarno Mertoguno, Kevin Hamlen, and Nicholas Evancich
DARPA I2O
Funded at \$10,647,759, 7/1/2021-12/1/2025.
2. **D3: Debloating, Dialecting and Diversification for Attack Resilient Software with Real-time Constraints**
Taesoo Kim (PI); co-PI: Wenke Lee
Technology Innovation Institute Sole Proprietorship LLC
Funded at \$1,575,000, 9/1/2020-8/31/2023.
3. **HECTOR: Halting Emergent Computation Through Oracle Reinforcement**
Wenke Lee (PI); co-PIs: Sukarno Mertoguno (GTRI) and Clayton Kerce (GTRI)
DARPA I2O
Funded at \$1,000,000, 1/15/2020-7/15/2021.
4. **Interactive Editing Techniques for Subsetting and Dialecting Network Protocols**
Taesoo Kim (PI); co-PIs: Wenke Lee, Alex Orso, and Brendan Saltaformaggio
ONR
Funded at \$2,976,781, 8/1/2018-7/31/2021
5. **SaTC: CORE: Medium: Understanding and Fortifying Machine Learning Based Security Analytics**
Polo Chau (PI); co-PIs: Wenke Lee, Le Song, and Taesoo Kim
NSF
Funded at \$1,200,000, 8/1/2017-7/31/2021
6. **Comprehensive System Debloating via Path-Based Learning and Late-Stage OS Composition**
Wenke Lee (for Bill Harris) (PI); co-PIs: Bill Harris, Taesoo Kim, Alessandro Orso, and Santosh Pande
ONR
Funded at \$7,500,000, 8/1/2017-7/31/2022
7. **CyberCorps Scholarship-for-Service (SFS)**
Mustaque Ahamad (PI); co-PIs: Wenke Lee, Taesoo Kim, Sy Goodman, and Manos Antonakakis
NSF
Funded at \$5,000,000, 9/1/2016-10/31/2021.

8. **Multi-Layer Adaptive and Proactive Strategic Cyber Defense**
Radha Poovendran (PI, University of Washington); co-PIs: Wenke Lee (Georgia Tech) and Tamer Basar (UIUC)
ARO
Funded at \$1,000,000, 8/1/2016-7/31/2019.
9. **Intel Science & Technology Center for Adversary-Resilient Security Analytics (ISTC-ARSA)**
Wenke Lee (PI); co-PIs: Taesoo Kim, Polo Chau (CSE), and Le Song (CSE)
Funded as a \$1,500,000 gift.
10. **THEIA: Tagging and Tracking of Multi-Level Host Events for Transparent Computing and Information Assurance**
Wenke Lee (PI); co-PIs: Taesoo Kim, Alex Orso, and Simon Chung
DARPA
Funded at \$4,193,126, 6/26/2015-6/26/2019.
11. **ADAPT: Analytical Framework for Actionable Defense against Advanced Persistent Threats**
Radha Poovendran (PI, University of Washington); co-PIs: Wenke Lee (Georgia Tech), Shankar Sastry and Anthony Joseph (UC Berkeley), Joao Hespanha (UCSB), Tamer Basar (UIUC), and Maryam Fazel and Linda Bushnell (University of Washington)
ONR MURI
Funded at \$7,500,000, 6/1/2016-5/31/2021.
12. **Embedasploit: A “Pen-Test in a Box” for Industrial Control Systems**
Wenke Lee (PI); co-PIs: Sal Stolfo at Columbia, Brendan Dolan-Gavitt at NYU Tandon School of Engineering
Office of Naval Research
Funded at \$1,250,351, 5/1/2015-4/30/2018.
13. **BFT++: Attack Tolerance in Hard Real-Time Systems**
Taesoo Kim (PI); co-PI: Wenke Lee
Office of Naval Research
Funded at \$1,245,720, 4/1/2015-3/31/2018.
14. **TWC SBE: TTP Option: Medium: Collaborative: EPICA: Empowering People to Overcome Information Controls and Attacks**
Wenke Lee (PI); co-PIs: Nick Feamster, Hongyuan Zha, and Hans Klein
NSF
Funded at \$1,100,000, 8/1/14-7/31/17.
15. **Comprehensive Understanding of Malicious Overlay Networks**
Homeland Security Advanced Project Research Agency (HSARPA of DHS)
Wenke Lee (PI); co-PIs: David Dagon at Georgia Tech, Chris Smoak of GTRI, Roberto Perdisci (UGA), April Lorenzen of Dissect Cyber, Paul Vixie of Internet Systems Consortium, Jody Westby of Global Cyber Risk, and Matt Jonkman of Open Information Security

Foundation.

Funded at \$1,873,078 for project total, with \$673,078 to CoC/GT, 10/1/12–9/30/15.

16. EAGER: The Conceptual Landscape of Information Manipulation

Wenke Lee (PI); co-PIs: Nick Feamster, and Hans Klein (Public Policy)

NSF

Funded at \$200,000, 8/1/12–7/31/14.

17. Integrating Security Concepts in an Undergraduate Computer Sciences Curriculum

Intel

Wenke Lee (PI); co-PIs: Mustaque Ahamad, Nick Feamster, and Paul Royal

Funded as unrestricted gift at \$50,000, 11/2011;

additional, unrestricted gift at \$35,000, 3/2012;

additional, unrestricted gift at \$50,000, 11/2012;

a total of \$135,000.

18. Monitoring Free and Open Access to Information on the Internet

Google

Nick Feamster and Wenke Lee (co-PIs)

Funded as unrestricted gift: \$1,000,000 base for 12/15/2010–12/14/2012;

\$500,000 option for 12/15/2012–12/14/2013;

additional gift at \$500,000, for 12/15/13–12/14/2014;

a total of \$2,000,000.

19. Dimensions of Network Intelligence

HSARPA of DHS

Wenke Lee (PI) and Paul Royal (co-PI)

Funded at \$1,022,000, 9/2012–9/2017.

20. Dynamic Reputation for DNS

Google

Wenke Lee (PI)

Funded as unrestricted gift at \$75,000, 7/2010.

21. PEASOUP: Preventing Exploits Against Software Of Uncertain Provenance

IARPA/STONESOUP Program (GrammaTech subcontract)

Wenke Lee (co-PI), with David Melski (PI) at GrammaTech, Jack Davidson and John Knight

(co-PIs) at University of Virginia, and Tom Bracewell (co-PI) at Raytheon

Funded at \$13,000,000 for project total, with \$1,695,887 to Georgia Tech, 6/1/2010–5/31/2014.

22. Research in Virtual Machine Monitoring Techniques

Sandia National Labs/Sandia Corp.

Wenke Lee (PI)

Funded at \$50,000, 9/15/2010–9/15/2011.

23. TC: SMALL: A Foundational and Practical Platform for Host Security Applications

NSF CNS/Trustworthy Computing

Wenke Lee (PI)

Funded at \$429,571, 7/15/2010–7/15/2013.

24. Botnet Attribution and Removal: From Axioms to Theory to Practice

ONR MURI

Wenke Lee (PI); Co-PIs: Nick Feamster, and Jon Giffin at Georgia Tech; Farnam Jahanian, Mike Bailey, and Kang Shin at University of Michigan, Ann Arbor; Giovanni Vigna and Chris Kruegel at University of California, Santa Barbara; John Mitchell at Stanford University
Funded: \$7,500,000 project total, \$2,600,000 for Georgia Tech, 5/1/2009–4/30/2014.

**25. Collaborative Research: CT-L: CLEANSE: Cross-Layer Large-Scale Efficient Analys
is of Network Activities to Secure the Internet**

NSF CNS/Cyber Trust

Wenke Lee (PI); Co-PIs: Nick Feamster, Jon Giffin, Mustaque Ahamad, and Xiaoming Huo (ISyE) at Georgia Tech; Mike Reiter and Fabian Monrose at University of North Carolina at Chapel Hill; Farnam Jahanian and Mike Bailey at University of Michigan; Phil Porras and Vinod Yegneswaran at SRI International; and Paul Vixie at Internet Software Consortium.
Funded: \$1,829,297 project total, \$758,297 for Georgia Tech, 9/1/2008–8/31/2012.

26. SMITE: Scalable Monitoring in the Extreme

DARPA

Wenke Lee and Nick Feamster (co-PIs at Georgia Tech); BBN subcontract
Funded: \$574,999 for Georgia Tech, 4/1/2008–3/31/2010

27. Countering Botnets: Anomaly-Based Detection, Comprehensive Analysis, and Efficient Mitigation

DHS HSARPA

Wenke Lee (PI); with Nick Feamster and Jon Giffin at Georgia Tech, David Dagon at Damballa, Rick Wesson and Adam Waters at Support Intelligence, Paul Vixie at Internet Systems Consortium, and Jody Westby at Global Cyber Risk
Funded: \$1,050,730 project total, \$300,000 for Georgia Tech, 4/1/2008–3/31/2010.
Add-on: \$305,066 project total, \$190,066 to Georgia Tech, 7/1/2009–9/30/2010.

28. CRI-IAD: Collaborative Research: Enabling Security and Network Management Research for Future Networks

NSF CRI

Nick Feamster (co-PI), Alex Gray, and Wenke Lee at Georgia Tech, and Z. Morley Mao (co-PI) and Farnam Jahanian at University of Michigan
Funded: \$397,426 for Georgia Tech (3/1/2008–2/28/2011)

29. Collaborative Research: CT-T: Logic and Data Flow Extraction for Live and Informed Malware Execution

NSF CNS/CyberTrust

Wenke Lee (co-PI); co-PIs: Paul Barford at University of Wisconsin at Madison, and Phil Porras at SRI International
Funded: \$220,000 for Georgia Tech (9/1/2007–8/31/2009)

30. **Foundational and Systems Support for Quantitative Trust Management**
Office of Naval Research MURI (University of Pennsylvania subcontract)
Sampath Kannan (PI); co-PIs: Insup Lee, Oleg Sokolsky, Matt Blazes, and Jonathan Smith at University of Pennsylvania, Wenke Lee at Georgia Tech, and Angelos Keromytis at Columbia University.
Funded: \$1,125,000 for Georgia Tech (project total: \$4,000,000) (5/1/2007–4/30/2012)
31. **Collaborative Research: CT-ISG: Modeling and Measuring Botnets**
NSF CNS/CyberTrust
Wenke Lee (PI) and Cliff C. Zou (co-PI) at University of Central Florida
Funded: \$350,000 for project total, Georgia Tech at \$175,000, (8/15/2006-8/15/2009)
32. **CT-ISG: Trusted Passage: Managing Distributed Trust to Meet the Needs of Emerging Applications**
NSF CNS/CyberTrust
Mustaque Ahamad (PI), Wenke Lee (co-PI), and Karsten Schwan (co-PI)
Funded: \$350,000, (8/15/2006-8/15/2009)
33. **Cyber Threat Analytics**
ARO via subcontract from SRI International
Funded: \$80,000, (7/15/2006-7/15/2007)
34. **ARL CTA**
ARL via subcontract from SPARTA/Telcordia
Funded: \$75,000, (7/15/2006-7/15/2007)
35. **Computer Network Defenses Technologies**
AFRL via subcontract from Scientific Systems Co. Funded: \$30,000, (9/13/2006-3/13/2007)
36. **ARO Workshop on Research in Botnets**
ARO
Funded: \$25,000, (4/1/2006-6/1/2007)
37. **Next-Generation Botnet Detection and Response**
DARPA (via ARO)
Funded: \$291,346, (12/15/2005-6/14/07)
38. **An Information-Theoretic Framework for Evaluating and Optimizing Intrusion Detection Performance**
Army Research Office
Funded: \$199,745, (4/1/2005-1/30/07)
39. **Preventing SQL Code Injection by Combining Static and Runtime Analysis, funded by Department of Homeland Security**
Department of Homeland Security
Alex Orso (PI) and Wenke Lee (co-PI)
Funded: \$200,000, (7/1/2005-12/31/2006)

40. **Anomaly and Misuse Detection in Network Traffic Streams – Checking and Machine Learning Approaches**
Office of Naval Research MURI (University of Pennsylvania subcontract)
Sampath Kannan (PI), Insup Lee, Oleg Sokolsky, and Linda Zhao at University of Pennsylvania, Wenke Lee at Georgia Tech, and Diana Spears and William Spears at University of Wyoming.
Funded: \$400,000 for Georgia Tech (\$2,000,000 project total) (7/15/2004–6/1/2006)
41. **Intrusion Detection and Security Management Technologies for Mobile Ad Hoc Networks**
Army Research Lab (Scientific Systems Company Inc. subcontract)
Funded: \$65,000 (2/1/2005-2/1/2006)
42. **A Course Module on Wireless Security**
Microsoft, Curriculum Development Grant
Funded: \$30,000 (unrestricted gift)
43. **Intrusion Detection Techniques for Mobile Ad Hoc Networks**
NSF, CISE/CCR/Trusted Computing
Funded: \$275,000 (8/15/2003-7/31/2006)
REU Funded: \$15,000 (9/10/2004-7/31/2006)
44. **High-Speed Intrusion Detection Sensors**
Army Research Lab (Scientific Systems Company Inc. subcontract)
Funded: \$65,000 (8/7/2003-8/6/2005)
45. **Distributed Intrusion Detection Feasibility Study**
U.S. Government (Columbia University subcontract)
Funded: \$45,000 (6/1/2003-6/30/2004)
46. **Agile Security for Storing Sensitive and Critical Information**
NSF, CISE/Trusted Computing
Mustaque Ahamad (PI), and co-PIs: Wenke Lee, Doug Blough, and H. Venkateswaran
Funded: \$460,000 (8/15/2002-7/31/2005)
47. **CAREER: Adaptive Intrusion Detection Systems**
NSF, CISE/CAREER
Funded: \$350,000 (8/15/2002–7/31/2007)
48. **ITR/SI: Guarding the Next Internet Frontier: Countering Denial of Information**
NSF, ITR
Mustaque Ahamad (PI), and co-PIs: Wenke Lee, Andre dos Santos, Ling Liu, Leo Mark, Ed Omiecinski, and Calton Pu
Funded: \$1,694,769 (9/1/2001–8/31/2006)
49. **Vulnerability Assessment Tools for Complex Information Networks**
Army Research Office MURI (Harvard University subcontract)
Yu-Chi Ho (PI) at Harvard University, co-PIs: Avrom Pfeffer at Harvard University, Christos

G. Cassandras at Boston University, Wei-Bo Gong at University of Massachusetts, Amherst, and Wenke Lee at Georgia Tech

Funded: \$400,000 for Georgia Tech (\$2,000,000 project total) (5/1/2001–4/30/2006)

50. **A Data Mining Approach for Building Cost-sensitive and Light Intrusion Detection Models.**

DARPA, ATO

Wenke Lee (PI), and co-PIs: Salvatore J. Stolfo at Columbia University and Philip Chan at Florida Institute of Technology

Funded: \$854,000 for Georgia Tech (\$2,000,001 project total) (6/1/2000–8/31/2003)

I. Research Honors and Awards

1. IEEE Fellow, 2021.
2. ACM SIGSAC (Special Interest Group on Security, Audit and Control) Outstanding Innovation Award for “contributions to network and systems security, in particular, machine-learning based approaches to security analytics, including tackling intrusion and botnet detection”, 2019.
3. CCS Test of Time Award, The 25th ACM Conference on Computer and Communications Security (CCS), for the paper “Artem Dinaburg, Paul Royal, Monirul Sharif, and Wenke Lee. Ether: Malware Analysis via Hardware Virtualization Extensions” published in CCS 2008 that have had the greatest impact on security research and practice over the past decade, 2018.
4. ACM Fellow, 2017.
5. Distinguished Practical Paper Award, for the paper “Yanick Fratantonio, Chenxiong Qian, Pak Chung, and Wenke Lee. Cloak and Dagger: From Two Permissions to Complete Control of the UI Feedback Loop” in Proceedings of The 2017 IEEE Symposium on Security and Privacy, 2017.
6. Internet Defense Prize, awarded by Facebook and USENIX, for the paper “Byoungyoung Lee, Chengyu Song, Taesoo Kim, and Wenke Lee. Type Casting Verification: Stopping an Emerging Attack Vector” in Proceedings of The 24th USENIX Security Symposium, 2015.
7. Cyber Is A Global Sport Award (awarded to David Dagon and Wenke Lee), Department of Homeland Security, Science and Technology Directorate, Cyber Security Division, 2015.
8. Outstanding Achievement in Research Program Development Award, (awarded to Mustaque Ahamad, Wenke Lee, Paul Royal, and David Dagon), Georgia Institute of Technology, 2015.
9. Outstanding Community Service Award, the IEEE Technical Committee on Security and Privacy, 2013
10. Outstanding Faculty Leadership for the Development of GRAs Award, Georgia Institute of Technology, 2012.
11. Sigma Xi Faculty Best Paper Award, Georgia Institute of Technology, 2010.
12. Best Student Paper Award, for the paper “Monirul Sharif, Andrea Lanzi, Jon Giffin, and Wenke Lee. Automatic Reverse Engineering of Malware Emulators” in Proceedings of The 2009 IEEE Symposium on Security and Privacy, 2009.
13. Outstanding Senior Faculty Research Award, College of Computing, Georgia Institute of Technology, 2009.
14. NSF CAREER Award, 2002.

15. Best Paper Award, Applied Research Category, the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '99), August 1999 (with Kui Mok and Sal Stolfo).
16. Honorable mention (runner-up) for Best Paper Award, Applied Research Category, the 4th International Conference on Knowledge Discovery and Data Mining (KDD '98), August 1998.
17. Honorable mention (runner-up) for Best Paper Award, Applied Research Category, the 3rd International Conference on Knowledge Discovery and Data Mining (KDD '97), August 1997.

III. SERVICE

A. Professional Activities

A.1 Membership and Activities in Professional Societies

1. Fellow, Association for Computing Machinery (ACM)
2. Fellow, Institute of Electrical and Electronics Engineers (IEEE)

A.2 Journal Editorial Board Activities

1. IEEE Transactions on Dependable and Secure Computing (TDSC) 2010 - 2011.
2. ACM Transactions on Information and System Security (TISSEC) 2005 - 2011.

A.3 Conference Committee Activities

1. The 31st USENIX Security Symposium (2022).
2. The 2020 ACM Conference on Computer and Communications Security.
3. The 2020 IEEE Symposium on Security & Privacy.
4. The 30th USENIX Security Symposium (2021).
5. The 2019 ACM Conference on Computer and Communications Security.
6. The 2019 IEEE Symposium on Security & Privacy.
7. The 2018 ACM Conference on Computer and Communications Security.
8. The 2018 IEEE Symposium on Security & Privacy.
9. The 2018 IEEE European Symposium on Security & Privacy.
10. The 2017 ACM Conference on Computer and Communications Security.
11. The 26th USENIX Security Symposium (Security '17).
12. Co-chair, Security and Privacy Track, WWW 2017.
13. Annual Network and Distributed System Security Symposium (NDSS) 2017.
14. The 25th USENIX Security Symposium (Security '16).
15. The 23rd ACM Conference on Computer and Communications Security (CCS 2016).
16. The 2016 IEEE European Symposium on Security & Privacy.
17. Annual Network and Distributed System Security Symposium (NDSS) 2016.

18. The 22nd ACM Conference on Computer and Communications Security (CCS 2015).
19. Co-chair, Security and Privacy Track, WWW 2015.
20. Annual Network and Distributed System Security Symposium (NDSS) 2015.
21. European Symposium on Research in Computer Security (ESORICS) 2014.
22. The 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2014).
23. Security and Privacy Track, WWW 2014.
24. *PC Chair*, The 2013 IEEE Symposium on Security and Privacy.
25. The 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 2013).
26. The 20th ACM Conference on Computer and Communications Security (CCS 2013).
27. The 16th International Symposium on Recent Advances in Intrusion Detection (RAID 2013).
28. The 19th ACM Conference on Computer and Communications Security (CCS 2012).
29. The 9th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DMIVA 2012).
30. The 7th USENIX Workshop on Hot Topics in Security (HotSec '12).
31. *PC co-Chair*, The 2012 IEEE Symposium on Security and Privacy.
32. Financial Cryptography and Data Security (FC 2012).
33. The 19th Annual Network and Distributed System Security Symposium (NDSS 2012).
34. The 8th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2011).
35. The 41st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2011).
36. The 20th USENIX Security Symposium (Security '11).
37. The 6th USENIX Workshop on Hot Topics in Security (HotSec '11).
38. The 2011 IEEE Symposium on Security and Privacy.
39. The 13th International Symposium on Recent Advances in Intrusion Detection (RAID 2010).
40. The 15th European Symposium on Research in Computer Security (ESORICS 2010).

41. The 7th Conference on Detection of Intrusions and Malware & Vulnerability Assessment (DIMVA 2010).
42. Security and Privacy Track, ICDCS 2010.
43. The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010).
44. Security and Privacy Track, WWW 2010.
45. *PC Chair*, The 17th Annual Network and Distributed System Security Symposium (NDSS 2010).
46. Financial Cryptography and Data Security (FC 2010).
47. The 25th Annual Computer Security Applications Conference (ACSAC 2009).
48. The 16th ACM Conference on Computer and Communications Security (CCS 2009).
49. 14th European Symposium on Research in Computer Security (ESORICS 2009).
50. The 18th USENIX Security Symposium (Security '09).
51. *PC Chair*, The Second USENIX Workshop on Large-Scale Exploits and Emergent Threats (LEET), 2009.
52. *PC Co-Chair*, The Second ACM Conference on Wireless Network Security (WiSec), 2009.
53. The 2009 IEEE Symposium on Security and Privacy.
54. The 16th Annual Network and Distributed System Security Symposium (NDSS 2009).
55. The 24th Annual Computer Security Applications Conference (ACSAC 2008).
56. Internet Measurement Conference (IMC) 2008.
57. 13th European Symposium on Research in Computer Security (ESORICS 2008).
58. The 15th ACM Conference on Computer and Communications Security (CCS 2008).
59. The 17th USENIX Security Symposium (Security '08).
60. 3rd USENIX Workshop on Hot Topics in Security (HotSec '08).
61. The 2008 IEEE Symposium on Security and Privacy.
62. The 15th Annual Network and Distributed System Security Symposium (NDSS 2008).
63. The 23rd Annual Computer Security Applications Conference (ACSAC 2007).
64. The 14th ACM Conference on Computer and Communications Security (CCS 2007).
65. The 10th International Symposium on Recent Advances in Intrusion Detection (RAID 2007).

66. The 16th USENIX Security Symposium (Security '07).
67. The 2007 IEEE Symposium on Security and Privacy.
68. The 14th Annual Network and Distributed System Security Symposium (NDSS 2007).
69. The 2nd International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2006).
70. The 15th USENIX Security Symposium (Security '06).
71. The 13th Annual Network and Distributed System Security Symposium (NDSS 2006).
72. The 1st International Conference on Security and Privacy for Emerging Areas in Communication Networks (SecureComm 2005).
73. The 2005 IEEE Symposium on Security and Privacy.
74. The 2005 ACM Workshop on Wireless Security (WiSe 2005).
75. The 2005 IEEE International Conference on Data Mining (ICDM 2005).
76. The Fourteenth International World Wide Web Conference (WWW 2005).
77. The 2005 International Conference on Distributed Computing Systems (ICDCS).
78. The 2004 ACM Workshop on Visualization and Data Mining for Computer Security (VizSEC/DMSEC 2004).
79. The 2004 ACM Workshop on Security of Ad Hoc and Sensor Networks (SASN 2004).
80. The 2004 ACM Workshop on Wireless Security (WiSe 2004).
81. Steering Committee and the 7th International Symposium on Recent Advances in Intrusion Detection (RAID 2004).
82. The 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2004.
83. The 2004 SIAM International Conference on Data Mining, 2004.
84. The 2004 IEEE International Conference on Data Mining (ICDM 2004).
85. The 2004 IEEE Information Assurance Workshop.
86. Co-Chair of the the ICDM Workshop on Data Mining for Computer Security (DMSEC 2003).
87. The 2003 IEEE International Conference on Data Mining (ICDM 2003).
88. The 2003 ACM Workshop on Wireless Security (WiSe 2003).

89. Steering Committee and the 6th International Symposium on Recent Advances in Intrusion Detection (RAID 2003).
90. The 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003.
91. The 2003 IEEE Information Assurance Workshop.
92. Steering Committee for the 5th International Symposium on Recent Advances in Intrusion Detection (RAID 2002).
93. The 2002 ACM New Security Paradigms Workshop.
94. The 2002 IEEE Symposium on Security and Privacy.
95. *PC Co-Chair*, The 4th International Symposium on Recent Advances in Intrusion Detection (RAID 2001).
96. Program Committee and Organizational Committee for the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000.

B. On-campus Georgia Tech Committees

1. 5-Year Review Committee for the Chair of School of Computer Science, College of Computing, 2016.
2. Faculty Recruiting Committee, School of Computer Science, 2014.
3. Faculty Recruiting Committee, School of Computer Science, 2012-13.
4. RPT Committee, College of Computing, 2010-2012.
5. College of Computing Dean Search Committee, 2009-2010, College of Computing.
6. Chair, Faculty Recruiting Committee, 2008-2009, School of Computer Science, College of Computing.
7. Chair Advisory Committee, 2008-2009, School of Computer Science, College of Computing.
8. Co-Chair, Faculty Recruiting Committee, 2007-2008, School of Computer Science, College of Computing.
9. RPT Executive Committee, 2007-2008, College of Computing.
10. Faculty Recruiting Committee, 2006-2007, College of Computing.
11. Faculty Recruiting Committee, 2005-2006, College of Computing.
12. Undergraduate Curriculum Task Force, 2004-2005, College of Computing.
13. Ph.D. Admissions Committee, 2003-2004, College of Computing.

14. Faculty Recruiting Committee, 2001-2003, College of Computing.

E. Consulting and Advisory Appointments

1. Advisory Board of the NUS-Singtel Cyber Security Research & Development Laboratory, National University of Singapore. 2019-2021.
2. Secure, Accessible & Fair Elections (SAFE) Commission, State of Georgia, 2018-2019
3. Advisory Board of the Faculty of Engineering, The Chinese University of Hong Kong. 2017-
4. Board of Trustees, Pace Academy. 2017-

IV. NATIONAL AND INTERNATIONAL PROFESSIONAL RECOGNITION

A. Patents

1. “Systems and methods for using video for user and message authentication”. Simon Chung, Wenke Lee, and Yeongjin Jang. U.S. Patent Number: 10,476,888, November 2019.
2. “Method and System for Detecting Malware”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 10,257,212, April 2019.
3. “Methods and systems for detecting compromised computers”, David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 10,044,748, August 2018.
4. “Method and system for detecting malicious and/or botnet-related domain names”. Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 10,027,688, July 2018.
5. “Method and system for network-based detecting of malware from behavioral clustering”. Roberto Perdisci and Wenke Lee. U.S. Patent Number: 9,948,671, April 2018.
6. “Method and system for detecting DGA-based malware”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 9,922,190, March 2018.
7. “Method and system for detecting malicious domain names at an upper DNS hierarchy”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 9,686,291, June 2017.
8. “Systems and methods of safeguarding user information while interacting with online service providers”. Wenke Lee, Sasha Boldyreva, Simon Chung, Billy Lau, and Chengyu Song. U.S. Patent Number: 9,659,189, May 2017.
9. “Method and System for Detecting Malware”. Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 9,525,699, December 2016.
10. “Method and system for determining whether domain names are legitimate or malicious”, Manos Antonakakis, Roberto Perdisci, David Dagon, and Wenke Lee. U.S. Patent Number: 9,516,058, December 2016.
11. “Method and systems for detecting compromised networks and/or computers”, David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 9,306,969, April 2016.
12. “Systems and methods for secure in-VM monitoring”, Monirul Sharif and Wenke Lee. U.S. Patent Number: 9,129,106, September 2015.
13. “Method and system for network-based detecting of malware from behavioral clustering”. Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 8,826,438, September 2014.

14. "Method and system for detecting malicious domain names at an upper DNS hierarchy". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Nikolaos Vasiloglou. U.S. Patent Number: 8,631,489, January 2014.
15. "Method and System for Detecting Malware". Manos Antonakakis, Roberto Perdisci, Wenke Lee, and Gunter Ollmann. U.S. Patent Number: 8,578,497, November 2013.
16. "Method and System for Detecting and Responding to Attacking Networks". David Dagon, Nick Feamster, Wenke Lee, Robert Edmonds, Richard Lipton, and Anirudh Ramachandran. U.S. Patent Number: 8,566,928, October 2013.
17. "Methods for Cost-Sensitive Modeling for Intrusion Detection and Response". Wei Fan, Wenke Lee, Matt Miller, and Sal Stolfo. U.S. Patent Number: 7,818,797, October 2010.
18. "Method and System for Using Intelligent Agents for Financial Transactions, Services, Accounting, and Advice". Dan Schutzer, Will Foster, Huanrui Hu, Wenke Lee, Sal Stolfo, and Wei Fan. U.S. Patent Number: 5,920,848, July 1999.

V. OTHER CONTRIBUTIONS

A. Seminar Presentations

1. “Machine Learning and Security: The Good, The Bad, and The Ugly”, Distinguished Lecture at Dartmouth College, April, 2021.
2. “Machine Learning and Security: The Good, The Bad, and The Ugly”, Distinguished Lecture at Ohio State University, January, 2021.
3. “Machine Learning and Security: The Good, The Bad, and The Ugly”, Keynote at the 2020 ACM Conference on Computer and Communications Security, November, 2021.
4. “Security Overlay”, Stony Brook University, March 2016.
5. “Security Overlay”, Princeton University, October 2015.
6. “Internet Monitoring via DNS Traffic Analysis”, Google-UMD Cybersecurity Seminar, November 2012.
7. “Botnet Detection and Response: Challenges and Opportunities”, Carnegie Mellon University, September 2007.
8. “Botnet Detection and Response”, Indiana University, March 2007.
9. “Botnet Detection and Response”, University of Pittsburgh, October 2006.
10. “Architecture Considerations for Anomaly Detection”, University of Texas at San Antonio, November, 2005
11. “Advanced Intrusion Detection Techniques”, National Security Research Institute, Daejeon, South Korea, February, 2005
12. “U.S. Policy and Recent Detection Techniques for Cyber Threats and Attacks”, National Cyber Security Center, Seoul South Korea, February, 2005
13. “Architecture Considerations for Anomaly Detection”, Purdue University, February, 2005
14. “Architecture Considerations for Anomaly Detection”, Graduate Center of The City University of New York, February, 2005
15. “Architecture Considerations for Anomaly Detection”, The University of Alabama, November, 2004
16. “Architecture Considerations for Anomaly Detection”, Carnegie Mellon University, September, 2004
17. “Architecture Considerations for Anomaly Detection”, University of Wisconsin at Madison, August, 2004

18. "Local Worm Detection and Response: Algorithms and Analytical Models", Lawrence Livermore National Laboratory, June 2004
19. "Intrusion Detection", Carnegie Mellon University, October 2003
20. "Algorithms for Recognizing New Attack Step Relationships", University of Pennsylvania, September 2003
21. "Intrusion Detection", Emory University, October 2001
22. "Developing Data Mining Techniques for Intrusion Detection: A Progress Report", Center for Education and Research in Information Assurance and Security, Purdue University, October 2000
23. "A Data Mining Framework for Building Intrusion Detection Models", Institute for Security Technology Studies, Dartmouth College, Hanover, NH, July 2000
24. "A Data Mining Framework for Building Intrusion Detection Models", Department of Electrical and Computer Engineering, University of Massachusetts, Amherst, MA, February 2000
25. "A Data Mining Framework for Building Intrusion Detection Models", HRL Laboratories, Malibu, CA, November 1999