UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT INC.
Patent Owner

Case No. IPR2023-00289
Patent No. 8,418,250

DECLARATION OF DR. WENKE LEE

TABLE OF CONTENTS

I, Dr. 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. 8,418,250 (the "'250 Patent") (Ex. 1001). I understand that the Challenged Claims are claims 1-30 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. 8,418,250 ("*'250 Patent*");
- Exhibit 1002 – File History of the '250 Patent (the "*'250 File History*");
- Exhibit 1004 – U.S. Patent Publication No. 2005/0210035 to Kester ("*Kester*");
- Exhibit 1005 – U.S. Patent No. 7,225,343 to Honig ("*Honig*");
- Exhibit 1006 – U.S. Patent No. 7,594,272 to Kennedy ("*Kennedy*")
- Exhibit 1007 – Plaintiffs' Opposition Markman Brief, 22-cv-00243 WDTX, No. 98
- Exhibit 1010 – Defendants' Opening Markman Brief, 22-cv-00243 WDTX, No. 86

- Exhibit 1011 – *Intrusion Detection with Unlabeled Data Using Clustering* by Leonid Portnoy, et al. ("*Portnoy*")
- Exhibit 1012 – U.S. Patent No. 6,944,772 to Dozortsev ("*Dozortsev*")
- Exhibit 1013 – U.S. Patent No. 6,772,363 to Chess et al. ("*Chess*")
- Exhibit 1014 – WO 2002/033525 to Shyne-Song Chuang
- Exhibit 1015 – EP 1,549,012 to Kristof De Spiegeleer
- Exhibit 1016 – EP 1,280,040 to Alexander James Hinchliffe, et al.
- Exhibit 1017 – U.S. Patent Publication No. 2004/0153644 to McCorkendale ("*McCorkendale*")
- Exhibit 1018 – U.S. Patent No. 7,516476 to Kraemer ("*Kraemer*")
- Exhibit 1019 – Harold S. Javitz et al., The NIDES Statistical Component: Description and Justification" ("*The NIDES Statistical Component: Description and Justification*")
- Exhibit 1020 – U.S. Patent No. 7,448,084 to Apap et al. ("*Apap*")

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 '250 Patent

24. The '250 Patent (Ex. 1001) relates to systems and methods for "classifying a computer object as malware" which employs a community-based architecture illustrated below in Figure 1:

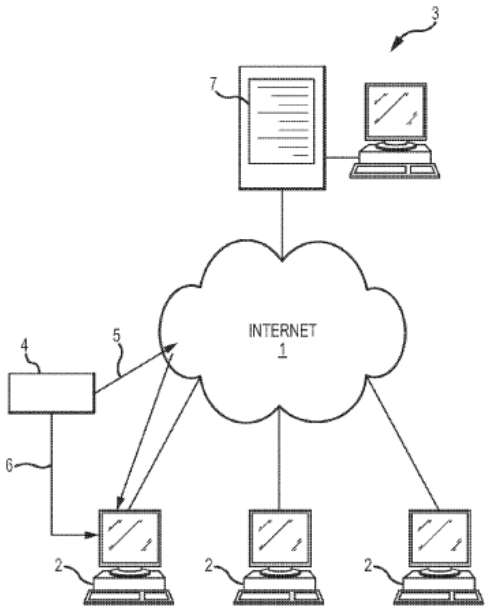


FIG.1

Id., Fig. 1, Abstract. The '250 Patent explains that these "'community-based' anti-malware systems in which a plurality of 'local' computer all connect via a network…to a central computer" have been long known in the art. *'250 Patent*, 1:60-2:27 (describing exemplary known in the art community-based anti-malware systems).

25. For example, the '250 Patent describes U.S. Patent Nos. 6,944,772; 6,772,346, and EP Patent Nos. 1549012; and 1280040. Each of these "disclose 'community-based' anti-malware systems" which operate as follows:

> On encountering a file that is not already known to them, the local computers send a request to the central computer for authorisation to run the file. If the file is recognised at the central computer, then the

> central computer can send permission for the local computer to run the file if the file is known to be safe or send a 'deny' command if the file is known to be malicious. However, in each of these prior art proposals, if the file is not known at the central computer, then the whole file is sent to the central computer where it can be analysed to determine whether it should be regarded as safe or malware.

Id., 1:60-2:7; *see also id*., 2:13-17 (noting the whole file can be sent to the central computer in the form of a "checksum"/"signature"/"key" that uniquely represents the file), 2:18-27 (describing additional community-based system in WO-A-02/33525 which performs a "manual" decision to run a file).

26. The '250 Patent explains that in these conventional community-based anti-malware systems, the analysis at the base computer "is typically carried out manually or semi-manually' by subjecting the file to detailed analysis…which can still take days given the hum involvement that is typically required." *Id*., 2:8-11. The "prior art systems either rely on deep analysis of a new object in order to determine whether or not the object is malicious, which introduces delay and therefore risk to users during the period that the file is analysed and new anti-malware signatures distributed, or limited analysis of the operation of the particular object or its method of transmission to a computer is carried out to decide a likelihood of the object being malicious." *Id*., 2:57-64.

27. In purported contrast to these prior art approaches that rely on maintaining a database of signatures (e.g., a hash or checksum) uniquely identifying

known malware for manual signature matching, the '250 Patent "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. Sophisticated pattern analysis can be carried out." *Id*., 3:8-17.

28. To accomplish this, as shown in Figure 1 above, the '250 Patent discloses a base computer that "holds a database 7 with which remote computers 2 can interact when the remote computers 2 run an object 4 to determine whether the object 4 is safe or unsafe. The community database 7 is populated, over time, with information relating to each object run on all of the connected remote computers 2." *Id*., 8:4-16. The "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." *Id*., Abstract, 4:31-42. This comparison/classification can involve exemplary heuristics or models that are used by the base computer to compare and classify the data, including analyzing patterns of behavior of an object. *Id*., 10:41-12:61 (describing exemplary heuristics), 13:59-64 (describing using behavior data "to model, test and create new automated rules" to determine a response to new or unknown activity).

29. More generally, the '250 Patent's techniques are "founded on receiving information about objects present on a plurality of remote computers at a base

computer" which "allows analysis to be performed mapping the behaviour and attributes of objects known across the community in order to identify suspicious behaviour and to identify malware at an early stage." *250 File History* (Ex. 1002), 466. This "method may comprise initially classifying an object as not malware, generating a mask for said object that defines acceptable behaviour for the object, and comprising [a] monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask." *'250 Patent*, 4:21-30.

30. In more detail, with respect to this mask, "[t]he signature of an object may comprise or be associated with a mask which can be built up with use of that object and which indicates the particular types of behaviour to be expected from the object." *Id*., 14:55-58. "The mask might indicate for example, the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication; any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on." *Id*., 14:61-15:2. In this way, "the mask can give an overall picture of the expected 'normal' behaviour of that object." *Id*., 15:2-4.

31. "[I]n one embodiment the behaviour of the object is continually monitored at the remote computer(s) 2 and information relating to that object continually sent to and from the community database 7 to determine whether the object is running within its expected mask." *Id.* at 15:5-8. "Any behvavior that extends beyond the mask is identified and can be used to continually assess whether the object continues to be safe or not." *Id*., 15:8-11; *see also id*., 15:11-25 (explaining behavior such as opening a new port to update itself or obtain regular data can be flagged and compared at other computers to see if any "ill effects" occurred from this behavior to mark as malware).

B. Overview of the '250 Patent's File History

32. I understand that the application, which issued as the '250 Patent, was filed on June 30, 2006 and claims priority to a foreign application filed June 30, 2005. I apply this June 30, 2005 date for the purposes of my opinions herein only as the priority date for the Challenged Claims.

33. I understand that the application that issued as the '250 Patent underwent several amendments to distinguish the claims from the prior art. For example, the Examiner rejected the claims over U.S. Patent Publication No. 2003/0177394 ("*Dozortsev*") in view of 2002/0099952 ("*Lambert*"). *'250 File History*, 565-571. The Examiner found that "Dozortsev does not explicitly teach generating a mask that defines acceptable behavior for the computer object, wherein

only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask," relying on *Lambert* for its teaching of the same. *Id.*, 566, 569.

34. In response, Applicant amended the independent claims to require "the received data to include information about the behaviour of the object, and the mask to be generated based on normal behaviour of the object determined from the received data (including behaviour information)." *Id*., 584. Applicant argued that, in *Lambert*, "[t]here is no teaching or suggestion of receiving behaviour information and using normal behaviour to define a mask" and that because "an administrator classifies the software, … [t]his is clearly manual, and not automatic generation of a mask." *Id.* at 585-586 (citations omitted).

35. The Examiner then rejected the claims over *Dozortsev* and *Kester.* The Examiner found "Dozortsev does not explicitly teach receiving data about the behavior of the object running on the remote computer, automatically generating a mask in accordance with normal behavior of the object determined from the received data that defines acceptable behavior for the computer object, and only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask." *Id.*, 611. Instead,

the Examiner found that these limitations were taught by *Kester*, and particularly, *Kester's* disclosure of a "workstation management module monitoring the behavior of an application," "wherein the behavior of the application is used to determine the access privileges for the application" and "that the privileges are used to determine what actions are permitted." *Id.* (*citing Kester* (Ex. 1004), [0041], [0043]-[0044], [0055]-[0056]).

36. In response, Applicant contended that "in Kester the hash/policy table 204 for each workstation stores details of previously seen programs (categorized programs) and previously seen network access requests (expected network access behavior) and policies for those programs and requests that have been set by the Network Administrator/human reviewer." *Id.*, 633. Applicant also alleged that *Kester* does not teach or suggest "reclassifying an object as malware if its monitored behavior extends beyond that permitted by the mask" and that "Kester has no teachings relating to classifying an object as malware at all." *Id*. According to Applicant, "[t]his is clearly different from the claims of the present invention" because the claimed process requires "a mask is automatically generated defining acceptable behavior for the object based on the normal behavior of the object. The automatically generated mask is used to automatically monitor the operation of the object and lo reclassify the object as malware if the behavior extends beyond the

mask. A human decision is not necessary to generate the mask or monitor the operation." *Id.* (emphasis original).

37. A notice of allowance then issued, explaining only that "applicant's arguments filed on December 23, 2011 are persuasive, as such the reasons for allowance are in all probability evident from the record." *Id.*, 803, 839. As I explain in detail in my opinions below, *Kester* in combination with *Honig* render obvious classifying/reclassifying an object as malware and automatically generating a mask.

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

38. 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 '250 Patent, which I have been instructed to assume is June 30, 2005. 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.

39. In my opinion, a person having ordinary skill in the art, as of June 30, 2005, 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 ’250 Patent and the prior art references discussed below.

40. 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 classification techniques described by the ’250 Patent. While the technology at issue in the ’250 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 ’250 Patent and the prior art at issue.

41. 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 ’250 Patent.

D. Background of the Technology

42. 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 June 30, 2005. As explained below, the centralized approach to identifying and classifying malware described by the ’250 Patent was developed in response to known inefficiencies in protecting networked computers from rapidly changing threats using hand-crafted, signature-based malware detection in the early 2000s and were thus well-known and conventional prior to the invention of the ’250 Patent.

43. The growing rate of interconnections among computer systems in the early 2000s has made network security a key focus for many researchers. To protect networks from exploitation, security system designers have long implemented Intrusion Detection Systems (IDSs). Increases in speed and complexity of computer networks, in addition to public access to these networks, have heightened the need

for effective intrusion detection approaches. The goal of intrusion detection is to build a system which automatically scans object or network activity and detects intrusion attacks. Once an attack is detected, the system and/or a system administrator can take corrective action. The two major categories for IDS's most often implemented are signature-based and anomaly-based detection.

44. With respect to signature based automatic detection methods, these methods were implemented to protect against widely known network intrusions. *Intrusion Detection with Unlabeled Data Using Clustering* by Leonid Portnoy, et al. ("*Portnoy*") (Ex. 1011), 1. These methods extract features from network data and detect intrusion by comparing the feature values to a set of known attack signatures often stored in a signature database. *Id*. However, these methods cannot detect new types of intrusions since they are limited to detecting known signatures. *Id*. These signature-based systems require system administrators to update the signature database for each new type of attack this is discovered. *Id*.

45. Aside from signature-based systems, anomaly-based approaches use data mining and machine learning algorithms to detect network intrusions. *Portnoy*, 1. There are two primary types of data mining/machine learning intrusion detection systems: (1) misuse detection; and (2) anomaly detection. *Id*., 1-2. In misuse detection, a set of data is labeled as either normal or an intrusion and a machine learning algorithm is trained over the labeled data. *Id*. For example, the known

MADAM/ID system extracts features from network connects and builds detection models over these connection records that represent of a summary of the traffic from a given connection. *Id*., 2. These detection models are generalized rules for classifying the data using extracted data. *Id*. However, this system requires a substantial magnitude of labeled data to accurately apply its rule-based approach. *Id*.

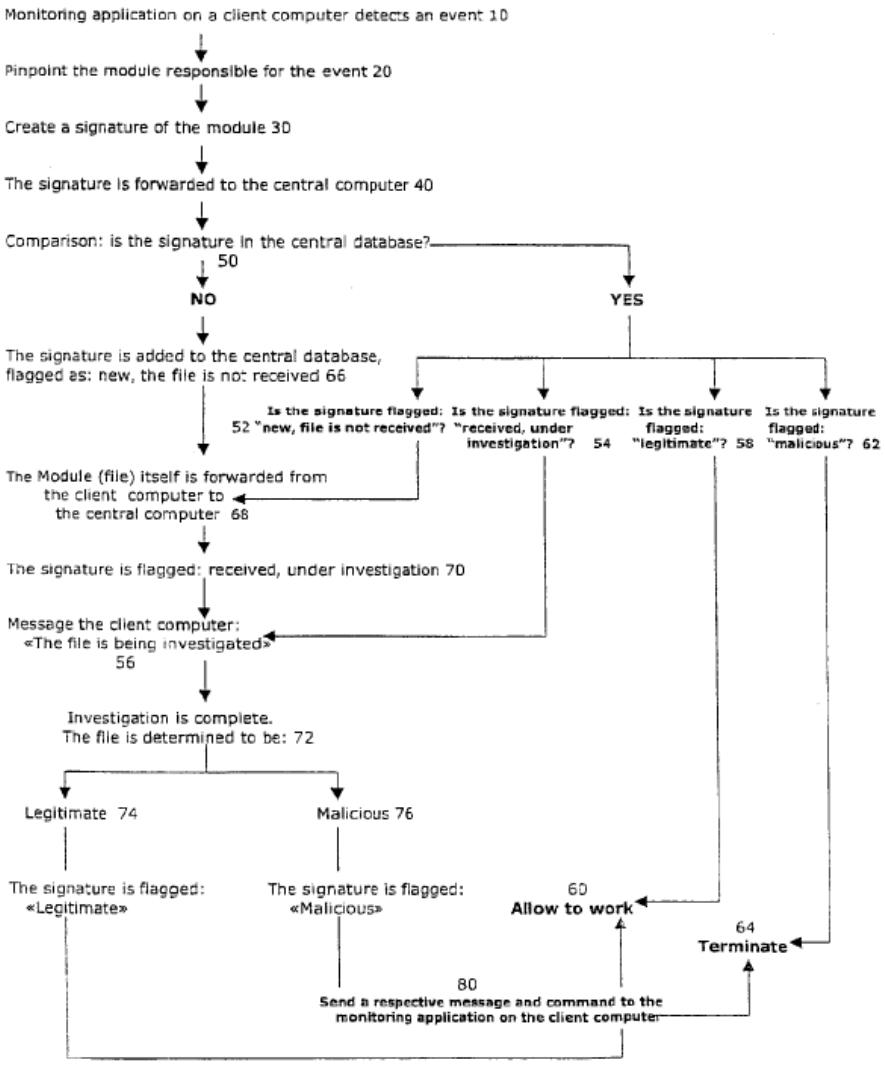
46. On the other hand, anomaly-based systems build models of normal network data and then detect deviations from this normal network data in observed network data. *Id*. These anomaly-based systems have the advantage that they can detect new types of intrusions, and do not require labeled data as an input. *Id*. These systems can be programmed to perform what is termed "unsupervised anomaly detection" in that a set of unlabeled data can be input into the system and the system can find intrusions buried within this data. *Id*. In the late 90's and early 00's, anomaly-based systems were the focus for many network security researchers.

47. Furthermore, understanding the limitations of conventional signature matching anti-malware techniques, industrial applications also evolved to implement community-based malware detection 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 behavioral 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, such as with the anomaly-detection methods discussed above.

48. An example community-based malware detection system is found in U.S. Patent No. 6,944,772 to Dozortsev ("*Dozortsev*") (Ex. 1012). 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*.

49. 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. 1013). 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. 1014), EP 1,549,012 to Kristof De

Spiegeleer (Ex. 1015), and EP 1,280,040 to Alexander James Hinchliffe, et al. (Ex. 1016) (all describing further community-based malware detection systems).

50. These community-based systems commonly employed heuristics or machine learning models to make a malware determination. For example, regarding heuristic-based approaches, U.S. Patent Publication No. 2004/0153644 to McCorkendale ("*McCorkendale*") (Ex. 1017) teaches execution authority 118 (i.e., a base computer) that uses heuristics to determine whether data about a piece of software received from plural computers (e.g., execution request frequency statistics) indicates that the software is a worm or otherwise malicious (malware). Namely, "malicious software detection module 512" of execution authority 118 includes heuristics module 520 that analyzes received software signatures "to identify characteristics of the software that are indicative of malicious software." *McCorkendale*, [0049]. In one embodiment, heuristics module 520 includes frequency monitoring module 522 that determines whether there is an "occurrence of an abnormally high frequency of requests from different client devices 122 to execute the same software within a relatively short time period," which "is indicative of a software worm trying to spread among the client devices 122 and thus suggests that the software is malicious." *Id.*, [0050]; *see also id.*, [0068], [0031], [0011]. For example, if the number of executions for a particular piece of software in any given hour exceeds a predetermined threshold, the module 522 determines that the

software is malicious. *Id.*, [0051]. Otherwise, the software is non-malicious and safe to execute. *Id*. This process is shown below in Figures 5 and 7:

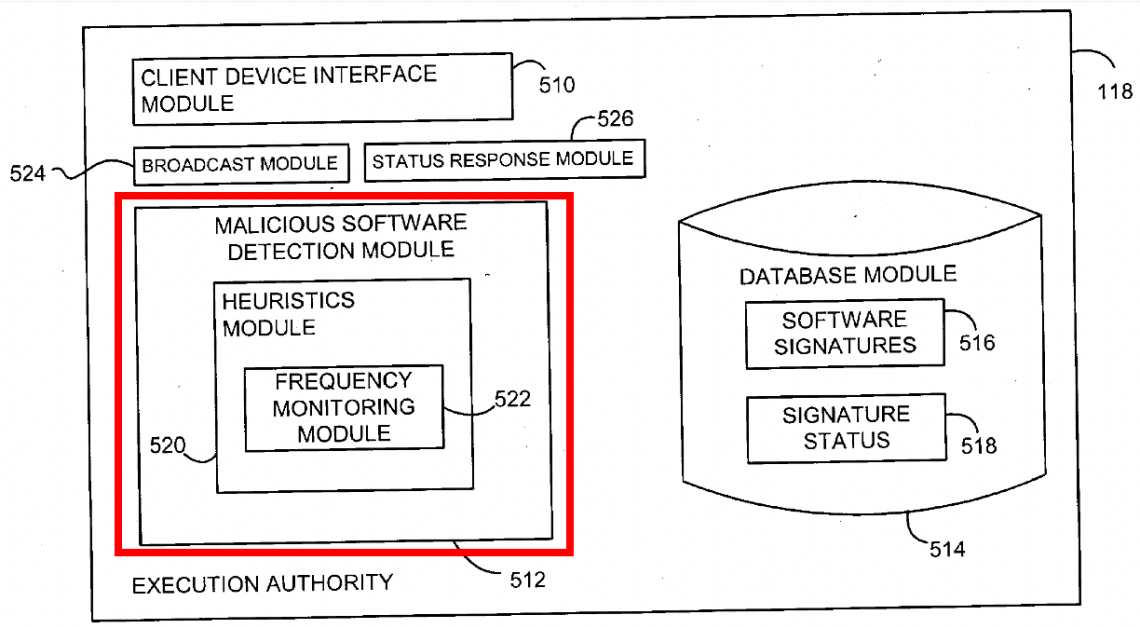


FIG. 5

McCorkendale, Fig. 5 (annotated).

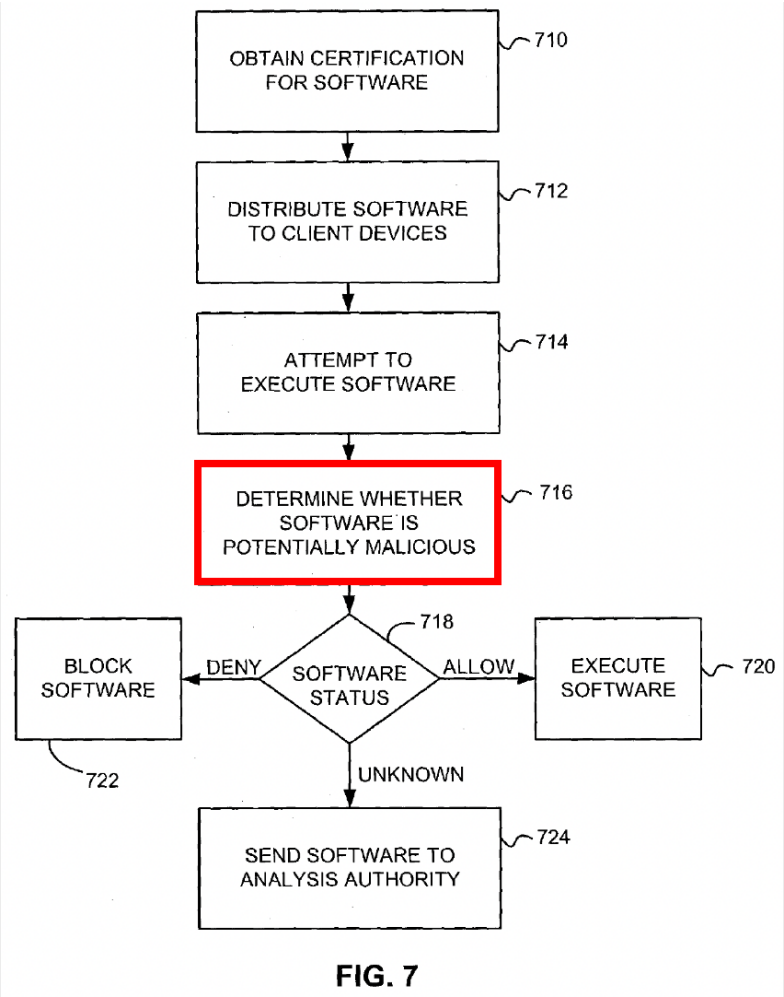


FIG. 7

McCorkendale, Fig. 7 (annotated).

51. Similarly, U.S. Patent No. 7,516476 to Kraemer ("*Kraemer*") (Ex. 1018) teaches an automated method for creating a security policy "to regulate the actions of one or more applications executing on a system, based on the observed behavior of the application(s)." *Kraemer*, 4:6-11. "Applications 125 may execute on one or more workstations on the system[.]" *Kraemer*, 5:35-36; *see also id.*, 6:46-56, 9:7-15. According to *Kraemer*, "[a]s the applications execute, the access requests they issue are monitored and recorded," such as using a reference monitor

executing on each workstation of the system. *Kraemer*, 4:19-21, 5:48-56. The gathered behavioral data includes information on application operations "wherein network resources are requested." *Kraemer*, 5:20-24, 5:57-6:3.

52. The gathered behavioral data (referred to as "analysis job data") is transferred to analysis workstation 130, where it is "aggregated into a set of 'representative' actions for the application(s) according to a heuristic." *Kraemer*, Abstract, 4:21-24, 6:57-7:3, Fig. 1. The heuristic, which may be configured with an automated routine, "defines the commonality between actions required to aggregate those actions into a representative action." *Kraemer*, 9:5-7, 9:28-45. The automated routine that configures the heuristic examines the behavioral data with an eye toward "a desired rigor" of the resulting security policy. *Id.* The representative actions are then used to develop rules of the security policy to regulate the behavior of the application(s) based on the previous behavior of the application(s). *Kraemer*, 4:6-11, 4:24-29, 9:46-10:24. *Kraemer* expressly states that the disclosed process of developing security policy, including aggregating the data into representative actions and developing a policy using the representative actions may be performed in a "completely automated fashion, ***wherein no human intervention is required***." *Kraemer*, 11:49-65.

53. In both heuristic and machine model approaches, a set of expected (i.e., "normal") activity is often compared to measured activity to determine if the

measured activity deviates from normal. Each of these processes are fully automated, such that no human involvement is required to generate normal behavior, compare it, and make the malware determination/classification. For example, and as explained in more detail in the Grounds below, *Honig* (Ex. 1005) discloses a model generator 16 that automatically generates a model of behavior for a computer object in accordance with normal behavior. *Honig*, 8:15-18 ("The data in the data warehouse 14 is accessed by the detection model generators 16 which generate models that classify activity as either malicious or normal."), 24:43-45 ("The model generator 16 then uses the unsupervised SVM algorithm to generate a model of normal activity"), 4:54-57 ("What is needed is an architecture to automate the processes of data collection, model generation and data analysis, and to solve many of the practical problems associated with the deployment of data mining-based IDSs."), 21:19-21, 6:50-63, 7:9-20, 19:46-59, 20:33-21:38.

54. *Honig* teaches three exemplary types of model generation algorithms supported by the system: "The first is misuse detection, which trains on labeled normal and attack data. The second is supervised (traditional) anomaly detection which trains on normal data. The third is unsupervised anomaly detection which trains on unlabeled data." *Honig*, 20:33-21:38. *Honig* teaches that each such model generation algorithm is "fully automated." *Honig*, 7:17-20 ("Automated model generation trains detection models from data stored in the data warehouse. The

process for converting the data into the appropriate form for training is fully automated."); *see also id*., 20:33-21:38.

E. Claim Construction

55. 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.

56. I have reviewed the claim construction discussion in the petition, which I understand proposes the Board apply Patent Owner's interpretation of the claims with the exception of "automatically." Specifically, I understand that Patent Owner has taken the position that the claimed method steps and/or claimed functions permit human involvement even where the claims recite that the operation must be performed "automatically." I understand that the petition proposes applying Patent Owner's interpretation that the claim limitations permit human involvement for all limitations other than those reciting "automatically." For those limitations that recite "automatically," pursuant to the petition's proposal, I have assumed that human action is excluded and that the automatic processes must be performed entirely

without human involvement. I take no position as to the correct interpretation of the claims with respect to human involvement.

F. Summary of the Prior Art References

1. *Kester*

57. *Kester* (Ex. 1004) provides a method and apparatus for classifying a computer object as malware based on the behavior of the computer object. In general, *Kester* teaches collecting network access behavior from multiple remote computers, referred to as "workstations," and using that data to determine an "expected or allowed network activity." "[T]he expected network activity for an application is determined from a record of that application's prior activity on multiple workstations" or it can be "determined from a record of that application's prior activity on the first workstation." The expected behavior is distributed to user workstations (remote computers) and each time the application attempts to access the network, the network access request is compared to the expected behavior. If it matches, network access is allowed, if it doesn't match, network access is denied and/or execution of the application is stopped.

58. *Kester's* process is implemented over the following community-based architecture:

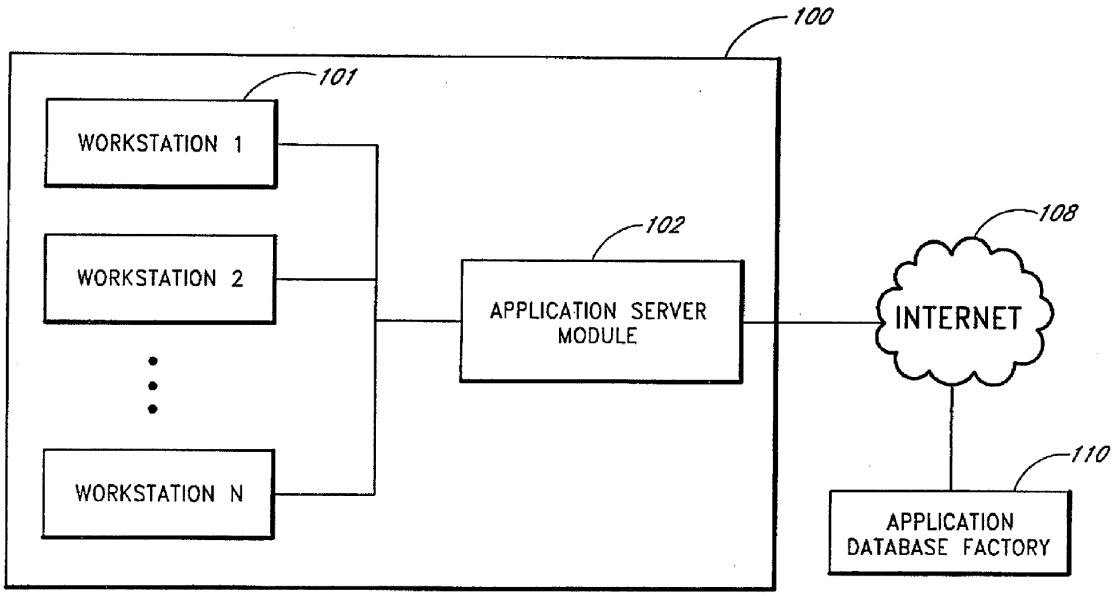


FIG. 1

Kester, Fig. 1. Specifically, *Kester's* workstations comprise a "workstation management module" configured to detect a network access request by an application. *Id.*; *see also id.*, Fig. 2. When the application accesses a network, *Kester's* system "monitors the behavior or activity of the application[.]" *Id.*, [0051] (referring to this behavioral information as "network access data"). This network access data "is uploaded to the application server module 102… [and] compared to expected network access data for the application." *Id.*, [0069]. The application server module "monitor[s] the behavior of categorized and uncategorized applications," and when a categorized application "does not operate in a predetermined manner [it] is placed in the uncategorized applications database 108." *Id.*, [0069]-[0070].

59. The applications and associated data placed in the uncategorized applications database are accessed by "[t]he network administrator, via the classification user interface 106, [which] is then able to categorize the uncategorized application and/or associate a policy with the category or application. Applications that the network administrator determines are not operating in a predetermined manner are categorized or re-categorized…. if the network administrator does not classify the application, the application database factory 110 can classify the collection data." *Id*., [0076]. The application analyst's classification module, a component of the application database factory, receives applications that have "not been previously categorized" or applications that are "not operating in a predetermined manner" along with "any additional data associated with the application[.]" *Id*., [0087]. "The human reviewer can interact with the application analyst classification module 302 via a graphical user interface (GUI). In this way, the GUI provides a graphical interface tool for the human reviewer to manipulate and manage the master application database 300… The GUI can include buttons preloaded with algorithmically derived hints to enhance productivity of the human reviewer." *Id*., [0090]. In addition to the "algorithmically derived hints" to determine the appropriate categories for the application, the "human reviewer uses the application, related information, and any Internet information to research the application… documents, specifications, manuals, and the like to best determine the

category or categories to associate with the application… [then] the human reviewer classifies each application using the evidence associated with the application, any hints from the related information, and/or other research." *Id*., [0135]; *see also id*., [0092] (noting categorization can also be accomplished with adaptive learning systems).

60. FIG. 4A illustrates a "database of parent groups and categories that are associated with the applications." *Id.,* [0081]. "[E]xemplary categories of applications include… **malicious applets and scripts**. The categories can be further grouped into parent groups… [which] might include… malware." *Id*., [0082]. "The application analyst classification module 302 can perform further categorization of the application. For example, in the parent group titled access/privacy, the application could be classified under… spy ware[.]" *Id*., [0098].

61. 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.

62. The claimed invention of the '250 Patent relates to a method and apparatus "for classifying a computer object as malware." *'250 Patent*, 1:4-5; *see also id.*, Claims 1, 13, and 25. As I described above, *Kester* also provides a method

and apparatus for classifying a computer object as malware. *See above*, ¶¶57-60; *see also*, *Kester*, [0134], [0098] (discussing "classifying application" into at least one parent group and/or category), Fig. 4A (including "Malware" parent group and "Malicious Applets and Scripts" category); *see also* my opinions below in Ground 1, Elements 1.2 and 1.3. *Kester* is therefore in the same field of endeavor as the '250 Patent.

63. The '250 Patent is also directed at the problem of initially classifying an object as not malware, generating a mask for the object that defines acceptable behavior for the object, and reclassifying the object if the actual monitored behavior is not normal—i.e., is extending beyond the mask. *'250* Patent, 4:21-30, 15:2-4. As described above, *Kester* is similarly directed to classifying applications, including malicious applets. *Kester*, [0082] ("The categories can be further grouped into parent groups. For example, parent groups might include… malware"). *Kester* also defines expected network behavior for the application and reclassifies the application when it does not operate in a predetermined manner. *Id.*, [0070], [0076], [0179]-[0181], Claims 1-2. *Kester* is therefore reasonably pertinent to a problem faced by the inventor.

2. *Honig*

64. *Honig* (Ex. 1005) identifies a need for "an architecture to **automate the processes of** data collection, model generation and data analysis, and to solve many

of the practical problems associated with the deployment of data mining-based IDSs," to improve Intrusion Detection Systems (IDS). *Honig,* 4:53-56. To address this need, a Support Vector Machine (SVM) type of model generation algorithm is implemented to perform Unsupervised Anomaly Detection. *Id.,* 21:39-41. "Unsupervised anomaly detection algorithms examine unlabeled data and attempt to detect intrusions buried within the unlabeled data. Unsupervised anomaly detection algorithms operate under the principle that intrusions are very rare compared to the normal data and they are also quantitatively different. Because of this, intrusions are outliers in the data and can be detected." *Id.*, 21:23-40. Thus, "[s]ince unsupervised anomaly detection can detect intrusions in an unlabeled data set, they are used inside the forensics analysis engines." *Id*., 21:29-31. "A forensic system 24 retrieves a set of historical data from the data warehouse 14... Once the data set is retrieved, the forensics analysis engine 24 may apply a detection algorithm to finds suspicious activity in the data set. The suspicious activity is then labeled (either anomalous or normal) using SQL statements to mark the appropriate data." *Id*., 15:13-52. Based on "the principle that attacks are behavior that is different from normal," a trained anomaly detection model "can then classify new data as normal or anomalous." (*Id.*, 21:8-12).

65. The claimed invention of the '250 Patent relates to a method and apparatus "for classifying a computer object as malware." *'250 Patent*, 1:4-5; *see*

also id., Claims 1, 13. Like the ’250 Patent, *Honig* teaches a method and system for classifying applications as normal or anomalous. *Honig*, 15:43-16:24 (describing an automated process to classify suspicious activity as “either anomalous” (i.e., malware) or normal); *see also id*., 8:14-18 (describing classifying activity as either “malicious or normal”). *Honig* is therefore in the same field of endeavor as the ’250 Patent.

66. The ’250 Patent also addresses the need to automatically generate (e.g., without human decisions) a mask for an object that defines normal/acceptable behavior for the object, and to classify the object as anomalous if the actual monitored behavior is not normal (extending beyond the mask). *’250 Patent*, 4:21-30, 15:2-4, Claims 1, 13, 25. Likewise, *Honig* is similarly directed to automatically generating models for classifying/reclassifying software as anomalous (malware) or normal and addresses the need to initially classify an object as not malware. *Honig*, 8:15-18 (“The data in the data warehouse 14 is accessed by the detection model generators 16 which generate models that classify activity as either malicious or normal”), 24:43-45 (“The model generator 16 then uses the unsupervised SVM algorithm to generate a model of normal activity.”); *see also id*., 4:54-57 (“What is needed is an architecture to automate the processes of … model generation … and to solve many of the practical problems associated with the deployment of data

mining-based IDSs."). *Honig* is therefore reasonably pertinent to a problem faced by the inventor.

3. *Kennedy*

67. *Kennedy* (Ex. 1006) is directed "in general to computer security and in particular to detection a computer worm and/or other type of malicious software," identifying "… a need in the art for a way to detect and block worms and other types of malicious software that does not suffer the drawbacks of current security software." *Kennedy*, 1:7-39. *Kennedy* meets this need by implementing a "malicious software detection module (MSDM) [that] monitors a storage device of the computer system for the arrival of software from a suspicious portal. The MSDM designates such software as suspicious. The MSDM tracks the set of files that are associated with the suspicious software. If the files in the set individually or collectively engage in Suspicious behavior, the MSDM declares the Suspicious software malicious and prevents file replication and/or other malicious behavior." *Id.*, Abstract, Fig. 4; *see also id*., 5:43-6:10.

68. The claimed invention of the *'250 Patent* relates to a method and apparatus "for classifying a computer object as malware." *'250 Patent* at 1:4-5; *see also id.* at Claims 1, 13. Like the '250 Patent, *Kennedy* teaches a malicious software detection module that includes a file set tracking and behavior module for declaring

software malicious. *Kennedy*, 1:6-9; *see also id*., Abstract, Fig. 4 (step 418). *Kennedy* is therefore in the same field of endeavor as the '250 Patent.

69. The *'250 Patent* also rapidly determines the nature of an object without requiring detailed analysis of the object itself or relying on signature matching, and includes a behavior module for declaring software malicious. *'250 Patent*, 2:57-3:18, 3:31-41, 1:9-11. Similarly, *Kennedy* is also reasonably pertinent to this problem, addressing a system which performs a rapid determination of the nature of an object, without requiring detailed analysis of the object itself or relying on signature matching. *Kennedy*, 1:9-52, 2:57-3:18, 3:31-41. *Kennedy* is therefore reasonably pertinent to a problem faced by the inventor.

G. Ground 1: Claims 1-2, 7-14, and 19-30 are obvious over *Kester* in view of *Honig*

1. *Claim 1*

(a) *1[P]. A method of classifying a computer object as malware, the method comprising:*

70. As I describe in my opinions below in Elements 1.2 and 1.3, *Kester* in view of *Honig* renders obvious a method of classifying a computer object as malware.

(b) *[1.1] at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;*

71. *Kester* describes an application server module in communication with an application database factory (i.e., *base computer*) that receives data about a program/application (i.e., *computer object*) from multiple workstations 101 (i.e., *each of plural remote computers*) on which the object is stored. *Kester*, [0008], *see also id.*, claim 1 ("The system comprises a workstation configured such that a program resident thereon can access a network, … send program data associated with the program to an application server module, … and an application server module coupled to the workstation and configured to receive the program data from the workstation management module … if the program is not operating in a predetermined manner, then send the program data to an application database factory"). This same application can be stored on the plurality of workstations 101 (i.e., *plural remote computers*). *Kester*, [0041], [0157] ("different workstations 101 can have…the same application running thereon").

72. This general architecture is illustrated below in annotated Figure 1:

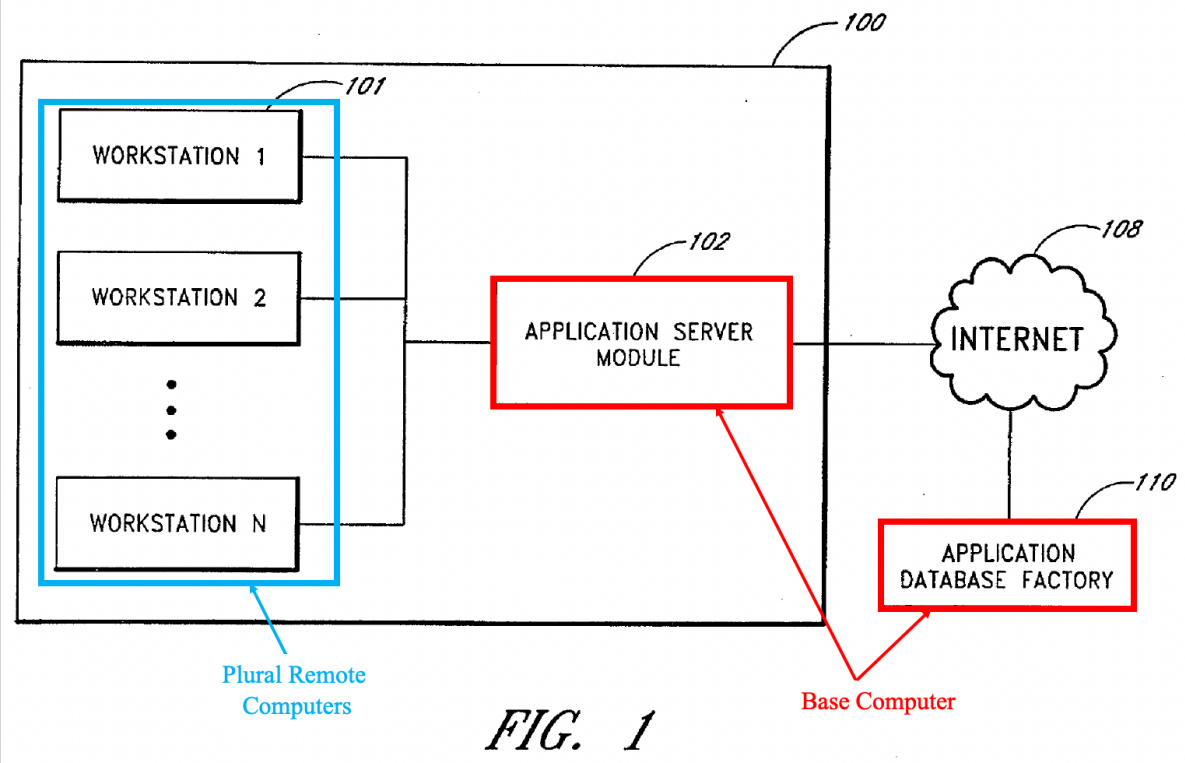


FIG. 1

Kester, Fig. 1 (annotated). As shown above, "[t]he application server module 102 communicates via the Internet 108 in order to upload and download applications and application related data with the application database factory 110." *Kester,* [0036], [0084].

73. In my opinion, one of ordinary skill in the art would have recognized the application server module and/or the application database factory of *Kester* as the "base computer" of the *'250 Patent*. The *'250 Patent* describes the functionality of the base computer as receiving details of the object "for storing in the community database 7 and preferably for further analysis at the base computer." *'250 Patent* at 8:44-47. As stated previously, *Kester* discloses uploading details of executed

applications to the application server module and to the application database factory. *Kester*, [0045] and [0069].

74. This received data (e.g., program/application related data, such as properties and attributes associated with the requested application) includes *information about the behavior of the program/application*—network access data; in addition to other information about the program/application from the remote workstations. For example, when an application on a workstation is launched and/or requests to access the network, the workstation logs application related data in logging database 206. *Kester*, Fig. 14 (steps 1402, 1412, 1418); *see generally id.*, [0139]-[0146], Figs. 2, 14. *Kester* uses the term "collection data" to refer to this application related data and additional data. *Id.*, [0073]. The collection data may include any information (e.g., properties/attributes) associated with the requested application including, but not limited to, behavior data (e.g., network access data) and other information about the application (e.g., application name, hash, publisher, suite, file size, directory location, execution request frequency). *Kester*, [0061]-[0062], [0073], [0051], [0099], [0141].

75. One of ordinary skill would have also understood that *Kester's* application-related data, referred to as "collection data," received by the application server module and/or application database factory that is transmitted from multiple workstations satisfies the claimed "*data including information about the behaviour*

of the object running on one or more remote computers" of the *'250 Patent*. *Kester's* collection data may include any information that was parsed or logged by the workstation, including "network access data" (*Kester*, [0073], [0076], [0099], [0112], [0141], [0147], [0150], [0153], [0166]), which establishes the "expected/predetermined behavior and unexpected behavior of the application." *Id.,* [0083], [0016], Fig. 4B. Likewise, the *'250 Patent* states that "[t]he data stored in the community database 7 provides an extensive corollary of an object's creation, configuration, execution, behaviour, identities and relationships to other objects that either act upon it or are acted upon by it." *'250 Patent*, 10:35-38.

76. Exemplary "network access data" (NET_ACTIVITY_INFO, shown in Figure 4B) includes information about the application and its behavior (transport protocol, destination port, and destination IP address)." *Id.,* [0083]; *see also id*., [0016], Fig. 4B:

```
<NET_ACTIVITY_INFO masked_hash="aafd61a161ae747844bf128d1b61747a95472570" transport="TCP"
port="80" ip="207.46.248.112" />
<NET_ACTIVITY_INFO masked_hash="473c07b881a1dc164206b95e61b8df20080d787c" transport="TCP"
port="80" ip="207.46.250.222" />
<NET_ACTIVITY_INFO masked_hash="473c07b881a1dc164206b95e61b8df20080d787c" transport="TCP"
port="443" ip="209.98.208.1" />
<NET_ACTIVITY_INFO masked_hash="c5e40cd2ee6c99a078e9b35e3a778f35d117clac" transport="UDP"
port="34047" ip="96.11.172.240" />
<NET_ACTIVITY_INFO masked_hash="4083b909153e7c1a2d38a39c08fc3912062b7196" transport="UDP"
port="2967" ip="128.0.0.109" />
<NET_ACTIVITY_INFO masked_hash="c5e40cd2ee6c99a078e9b35e3a778f35d117c1ac" transport="UDP"
port="7553" ip="96.219.68.240" />
```

FIG. 4B

77. After this data is logged, it is sent to the application server module and/or application database factory. For example, *Kester* discloses uploading program/application related data from multiple (potentially thousands) of workstations. *Kester*, [0119]. Each workstation uploads application related data from its logging database 206 to the application server module 102 "immediate[ly] or periodic[ally]." *Kester*, [0064]-[0065], Fig. 2; *see generally id*., [0073], [0020]-[0021], [0027]-[0028], Figs. 8, 9, 15, 16. This is illustrated below in annotated Figure 16:

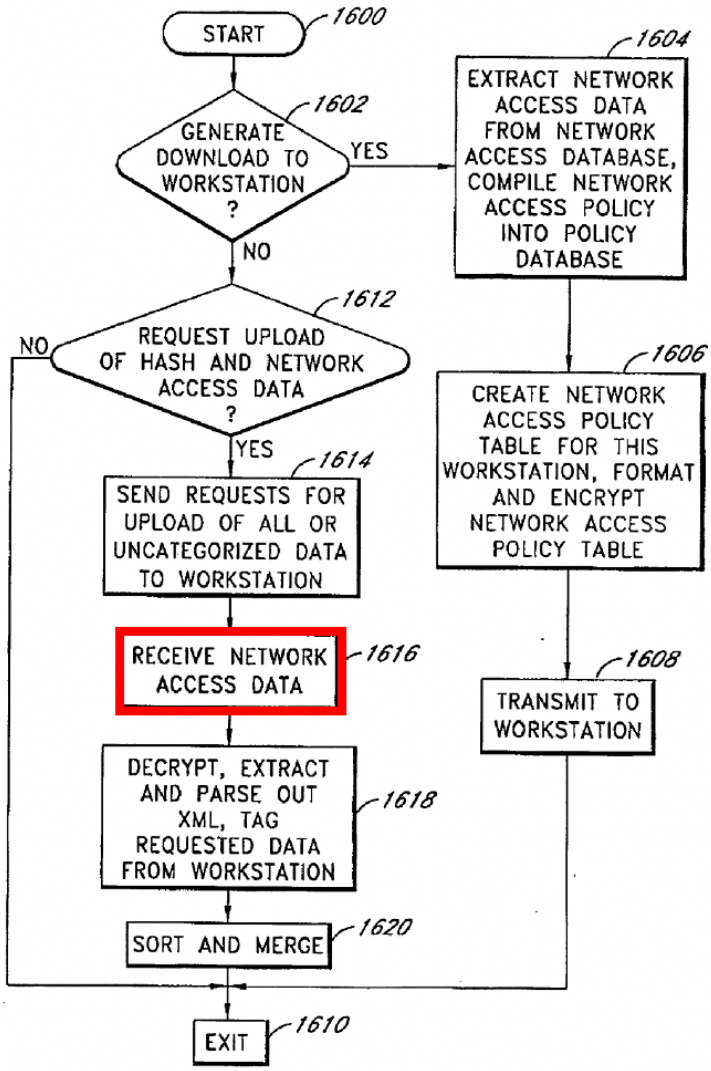


FIG. 16

Kester, Fig. 16 (annotated)

78. The application server module may then transmit application related data to the application database factory 110 "immediate[ly] or periodic[ally]." *Kester*, [0078], Fig. 3; *see generally id.*, [0086]-[0087], [0132], [0176], Figs. 11, 12, 18, 19. Exemplary application related data received by and displayed at the application database factory is illustrated in Figure 6. *Kester*, [0097]-[0098], Fig. 6.

79. For these reasons, one of ordinary skill would have understood that *Kester's* application server module and/or application database factory (*base computer*) receives collection data (*data about a computer object*) from each of plural workstations (*remote computers*) on which the object or similar objects are stored (*application*), the data including information about the behaviour of the object running on one or more remote computers (*network access data*).

(c) *[1.2] determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;*

80. *Kester* teaches determining in the application server module 102 and/or application factory database 110 (i.e., *base computer*) whether the application related data (i.e., *data about the computer object received from plural computers*) indicates that the application is malware.

81. Specifically, *Kester* describes a process where each application file can be associated with one or more parent groups and/or categories. Namely, when a categorized application does not operate in a predetermined manner, *Kester's* application server module programmatically places the application "in the uncategorized applications database 108." *Kester,* [0070]. "[T]he application database factory 110 receives uncategorized applications and any additional data associated with the application from the application server module 102...The application database factory 110 can include…an application analyst's classification

module 302." *Id*., [0084]-[0085]. "[T]he application analyst's classification module 302 receives the application and any additional data associated with the application [and] classifies or categorizes applications which are then added to the master application database 300 of categorized applications." *Id*., [0087]-[0089].

82. The application analyst's classification module "allows the human reviewer to analyze each application and any additional data that is associated with the application to determine its appropriate category or categories." *Id*., [0134]. *Kester* illustrates a database of parent groups and categories associated with the applications in Figure 4A and states that "exemplary categories of applications include… **malicious applets and scripts**. The categories can be further grouped into parent groups. For example, parent groups might include… **malware**." *Id.*, [0081]-[0082]. These parent groups and categories are illustrated in Figure 4A below, boxed in red:

FIG. 4A

Parent group	Category	Security VC	Liability VC	Productivity VC
System				
	Operating Systems			
	Device Drivers			
	File Management			
	Installers			
	Miscellaneous Utilities			
Access/Privacy				
	Anti-virus Software			
	Authentication			
	Encryption			
	Firewalls			
	Hacking	X	X	
	Remote Access			
	Spyware	X		
	System Audit			
Productivity				
	Contact Managers			
	CRM			
	Data Warehouse, Analytics, Reporting			
	Database			
	Document Viewers			
	ERP/SCM			
	Graphics			
	Infrastructure			
	Presentation			
	Project Managers			
	Proprietary			
	Reference			
	Search/Retrieval/Knowledge Management			
	Software Development			
	Spreadsheets			
	Suite/Integrated			
	Web/Desktop Publishing			
	Word Processing			
Communication				
	Collaboration			
	Dedicated Browsers			X
	Email			
	Instant Messaging			
	P2P Filesharing		X	
	Telephone/Conference/Fax			
	Web Browser			
Audio/Video				
	Media Players			X
	Imaging			X
Entertainment				
	Adult		X	
	Gambling		X	
	Games			X
Malware				
	Malicious Applets and Scripts			

Kester, Fig. 4A (annotated)

83. *Kester* states that the application analyst's classification module "display[s] the application and any related data on the GUI. The related data can indicate to the human reviewer the category or categories with which the application should be associated… [which] allows the human reviewer to analyze each application and any additional data that is associated with the application to determine its appropriate category or categories." *Id*., [0134]. Thus, one of ordinary skill in the art would have understood that the application and the application related data within the database factory indicates the appropriate category for an application.

84. A skilled artisan would have understood that an application classified into the "malicious applets and scripts" category, and/or the "malware" parent group involves determining whether this data indicates the application is malware. Indeed, the '250 Patent defines "malware" as including any object containing "malicious code" such as "viruses, Trojans, worms, spyware, adware, etc. and the like." *'250 Patent*, 1:14-17. Thus, *Kester's* determination of categories for a file such as "malicious scripts" falls squarely within this definition. As does a parent grouping for a file as "malware."

85. Indeed, it was well-known that abnormal behavior of a program or application indicates that the program or application may be malicious. For example, U.S. Patent Publication No. 2004/0153644 to McCorkendale ("*McCorkendale*") describes a similar malware detection system that detects abnormal behavior to

identify malware. Specifically, *McCorkendale* teaches an execution authority that "includes a computer system and contains functionality and information utilized by client devices 122 to prevent the execution of malicious software such as worms…The execution authority 118 monitors the software executions and utilizes execution frequency statistics to identify possible software worms." *McCorkendale*, [0031]. The execution frequency indicates that "software is potentially malicious upon the occurrence of an abnormally high frequency of requests from different client devices 122 to execute the same software within a relatively short time period.

> This high frequency of requests is indicative of a software worm trying to spread among the client devices 122 and thus suggests that the software is malicious. Similarly, an abnormally high frequency of requests from a single client device 122 to execute the same software may also indicate that the software is malicious.

Id., [0050].

86. By tracking "software execution frequencies over sliding time windows," *McCorkendale* determines that, when the "number of executions exceeds a predetermined threshold," the data indicates "that the software is malicious." *See Id*., [0051].

87. As stated above, according to the '250 Patent, malware "is used herein to refer generally to any executable computer file or, more generally 'object', that is

or contains malicious code, and thus includes viruses, Trojans, worms, spyware, adware, etc. and the like." *'250 Patent*, 1:14-17. Thus, *McCorkendale's* determination that received data indicates that the software is a worm or otherwise malicious is a determination that the received data indicates that the software is malware according to the *'250 Patent*. This is simply another known solution that is consistent with *Kester's* determination that the collection data indicates that an application is malware when it is determined that an appropriate category for an application, e.g., "malicious scripts," etc.

88. For these reasons, a person having ordinary skill in the art at the time of the invention would have understood that *Kester* teaches determining, in the base computer, whether the data about the computer object received from the plural computers indicates that the computer object is malware as defined by the '250 Patent.

Alternative Combined Teachings of Kester and Honig

89. Alternatively, it is my opinion that *Kester* in view of the teachings in *Honig* also renders this element obvious.

90. As I explained above in ¶¶ 55-56, I understand that Patent Owner, in parallel litigation, has argued that human intervention is not excluded for any claims. Thus, my opinions apply this view that no limitations exclude human involvement,

apart from Elements 1.5 and 1.7 which recite an "automatic[]" process, which I understand means "without human involvement."

91. While *Kester* states that a "network administrator, via the classification user interface 106, is [] able to categorize the uncategorized application" (*Kester,* [0076), and that "[a] human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization… via a graphical user interface (GUI)" (*id.*, [0089]-[0090]), *Kester* also states that the application analyst's classification module can "determine one or more appropriate categories… based upon word analysis, adaptive learning systems, and image analysis," (*Id.*, [0092]) and "perform further categorization of the application." *Id.,* [00098].

92. In addition to classifying applications by a human reviewer or network administrator, classifying applications without, or in addition to, human intervention was known in the art at the time of the claimed invention. One such example is described in *Honig*. *Honig* improves malware detection systems that audit network or system data "to look for clues of malicious behavior" by introducing "an architecture to **automate the processes of** data collection, model generation and data analysis, and to solve many of the practical problems associated with the deployment of data mining-based IDSs." *Honig,* 1:49-57, 4:53-56. *Honig* does this with detection model generators 16 that access data in a data warehouse and "**generate models that**

classify activity as either malicious or normal." *Id.*, 8:15-18. "The model generator 16 then uses the unsupervised SVM algorithm to generate a model of normal activity… Once the detection model is in place, sensors 12 send data to the detector 20 for classification." *Id*., 24:43-49; *see also id*., 8:4-24 (noting *Honig's* framework can "handle virtually any data type" as input, including "audit data").

93. *Honig* describes three exemplary types of model generation algorithms supported by the system that were known in the art at the time of disclosure: "The first is misuse detection, which trains on labeled normal and attack data. The second is supervised (traditional) anomaly detection which trains on normal data. The third is unsupervised anomaly detection which trains on unlabeled data." *Honig*, 20:33-21:38. *Honig* provides examples of known misuse detection algorithms and implementations (*id*., 2:29-42, 4:21-25) and known supervised (traditional) and unsupervised anomaly detection algorithms and implementations (*id*., 2:43-65, 3:42-4:47, 21:39-24:32). *Honig's* system uses Support Vector Machines (SVMs) as a type of model generation algorithm, which "can be used for both Unsupervised Anomaly Detection and normal Anomaly Detection[.]" *Id.*, 21:39-42. "The standard SVM algorithm is a supervised learning algorithm. It requires labeled training data to create its classification rule. An unsupervised variant does not require its training set to be labeled to determine a decision surface[.]" *Id.,* 23:33-40. The model generation algorithms are "**fully automated**." *Id.*, 7:17-20 ("Automated model generation

trains detection models from data stored in the data warehouse. The process for converting the data into the appropriate form for training is fully automated"); *see also id.*, 20:33-21:38.

94. One of ordinary skill in the art would have recognized that *Honig's* unsupervised SVM algorithm is an adaptive learning system that programmatically determines whether data is malware. *Honig's* "'Adaptive Model Generation' or AMG" architecture is a "framework [that] supports the creation of anomaly detection models" (*id*., 21:13-14) such that "any supervised learning module, any unsupervised (anomaly detection) learning model may be easily inserted into the architecture" (*id*., 7:38-41).

95. In my opinion, it would have been obvious to implement *Kester's* application categorization process, such that the application server module and/or database factory uses an adaptive learning system (rather than or in addition to a human reviewer) to determine whether received data indicates the application is malware, as taught by *Honig*. In the combined *Kester/Honig* method/system, *Kester's* server module and/or database factory would categorize applications using an adaptive learning model, which *Kester* already expressly contemplates. *See Kester*, [0092] ("Categorization can be based upon word analysis, ***adaptive learning systems***, and image analysis"). A skilled artisan would have understood this implementation would not preclude a human reviewer from validating and/or

modifying the automated determination, and therefore would not fundamentally change the overall operation of *Kester's* method.

96. One of skill in the art would have been motivated to implement the combined *Kester/Honig* method to improve *Kester's* malware detection system—such as by reducing human error associated with analyzing and classifying data. Not only does *Kester* disclose using adaptive learning systems to categorize applications, but *Kester* also includes a malware parent group and "malicious applets and scripts" and "spy ware" categories. *Kester,* [0082], [0098]. *Honig's* adaptive learning models for detecting malware "provide[] and automate[] many of the critical processes in the deployment and operation of real-time data mining-based intrusion detection systems," (*Honig,* 6:50-54). Implementing a learning model that provides and automates critical deployment processes, as disclosed by *Honig*, advances one of *Kester's* expressly stated goals to "enhance productivity of the human reviewer," thus motivating such implementation. *Kester*, [0090]. *Kester's* process would therefore be significantly improved through such automation, increasing reliability of results and reducing error margins. Indeed, *Honig's* models can review large volumes of data (including multi-dimensional data) and discover specific trends and patterns not apparent or not easily discoverable by human review. Learning models also adapt on the fly and learn through continuous use, making them more advantageous.

97. Moreover, a skilled artisan would have been motivated to configure *Kester's* server module and/or database factory to use an adaptive learning model to determine whether received application related data indicates that the application is malware as taught by *Honig* because this combination would have been a combination of prior art elements (*Kester's* adaptive learning system to determine malware and *Honig's* adaptive learning model to determine malware) according to known methods (*Honig's* express discussion that these concepts were well-known) to yield predictable results (an automated malware determination). *Kester's* application analyst classification module, which is part of the application database factory, implements adaptive learning systems to determine one or more appropriate categories for an application based on the application, collection data, and any additional data associated with the application. *Kester,* [0092]. *Honig's* adaptive learning models are also used classify activities or behaviors based on data collected within the system. *Honig*, 8:14-18. Incorporating *Honig's* adaptive learning models that perform classification as the adaptive learning system of *Kester's* application analyst classification module that also performs classification would have been a combination of known element according to known methods, the implementation of which would have been within the capabilities of the ordinarily skilled artisan. The combination would have predictably resulted in a system that determines *in the base*

computer whether the data about the computer object received from the plural computers indicates that the computer object is malware, as claimed.

98. As I described above, *Honig* explains that automating malware classification determinations through adaptive learning was known in the prior art. Such a modification to include well-known and widely employed concepts would have been well within the capabilities of one of ordinary skill in the art. A skilled artisan, therefore, would have appreciated a reasonable expectation of success in modifying *Kester's* application analyst classification module to include *Honig's* adaptive learning models for their stated purposes according to well-known methods, yielding no fundamental change in the overall operation of *Kester's* methods.

(d) *[1.3] classifying the computer object as malware when the data indicates that the computer object is malware;*

99. As I explained in my opinions above, *Kester* or alternatively *Kester* in view of *Honig* teaches determining an application to be malware when the application related data indicates the application to be malware and assigning an appropriate category/parent group to this application (e.g., "malicious scripts" or "malware") thus classifying this object as malware. Indeed, determining which group an application falls into includes *determining whether the object is malware*; and performing the step to assign that specific group to the application includes *classifying the application*.

100. Regarding *Kester's* classification process, *Kester* teaches two methods to classify applications into appropriate categories: one at the application server module; and one at the application database factory. For the former, *Kester* teaches that a human reviewer (e.g., network administrator) of the server module "classifies the application based on the displayed data," where the "displayed data" includes the application related data and hints (*see* Elements 1.1, 1.2). *Kester*, [0120], Fig. 10. For the latter, *Kester* teaches using an application analyst classification module 302 to classify the application into one or more "appropriate" categories. *See* Element 1.2; *Kester*, [0092], Claim 2

101. In more detail regarding classification at the application database factory, *Kester* discloses that "[t]he application analyst's classification module 302 classifies or categorizes applications which are then added to the master application database 300 of categorized applications. A human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization." *See Kester*, [0089]. *Kester* teaches that the classification module can analyze application related data "to determine the one or more appropriate categories," such as based on "adaptive learning systems." *See* Element 1.2; *Kester*, [0092] ("The application analyst classification module "analyze[s] each application, the collection data, any text objects associated with the application, any additional data associated with the application, and any additional data retrieved independent

of the collection data to determine one or more appropriate categories... Categorization can be based upon word analysis, adaptive learning systems, and image analysis").

102. The classification module user interface is shown in Figure 6 (below). The application related data for the "CMD PCI IDE Bus Driver" application includes the publisher (CMD Technology, Inc.), related suite (Microsoft Windows Operating System), filename, and directory location, indicating that the application should be classified in the "operating system" category. *Kester,* [0097]; Fig. 6:

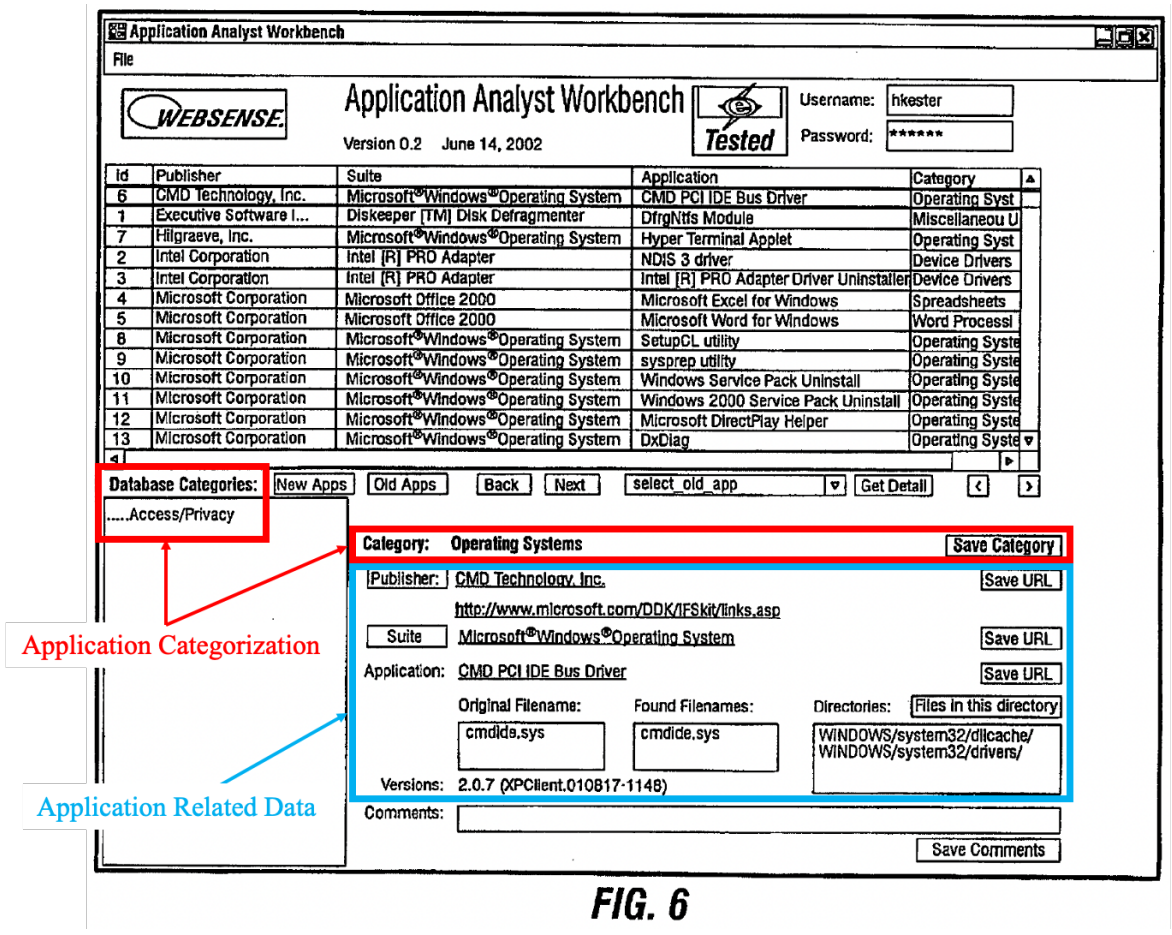


FIG. 6

Accordingly, "the application analyst's classification module 302 classified the application in the parent group titled access/privacy" and "operating system" category. *Kester,* [0098], Fig. 6; *see also Kester*, Claim 19 ("the application database factory comprises:…an application analyst's classification module configured to categorize the program").

103. In regard to Element 1.3, *Kester* expressly notes that "the application could be **classified** under … hacking, remote access, **spy ware**," among others. Similarly, "**malware**" is an alternative parent group and "**malicious applets and scripts**" an alternative associated category. *Kester*, [0082], Fig. 4A; *see also* Element 1.2.

Alternative Combined Teachings of Kester and Honig

104. Alternatively, it is my opinion that *Kester* in view of the teachings in *Honig* also renders this element obvious.

105. At the time of the invention, a person of skill in the art would have understood that human assistance is not required to classify an application into a category indicated by the application data. *Kester* contemplates that the application database factory can classify the application "if the network administrator does not classify the application." *See Kester*, [0076]; *see also id*., [0092] (describing autonomous processes, including adaptive learning models). While *Kester's* embodiments contemplate the validation and/or assistance of a human reviewer, as

I described above, it was well-known in the art to employ a "fully automated" system which can determine and classify data as malware. *See* Element 1.2 (describing *Honig's* adaptive learning model). A skilled artisan would have been motivated to implement *Kester's* application classification process, such that the application server module and/or database factory initially classify and/or reclassify applications into the appropriate group/category (e.g., malware) without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above.

106. As also briefly described above, another known in the art automated solution was described in *McCorkendale*. There, *McCorkendale* describes using heuristics to determine whether received application-related data (e.g., execution frequency statistics) indicates that a piece of software is malware and classifies it accordingly. Specifically, *McCorkendale* teaches that a "malicious software detection module 512" of execution authority 118 includes heuristics module 520 that analyzes received software signatures "to identify characteristics of the software that are indicative of malicious software." *McCorkendale*, [0049]. When the frequency monitoring module 522 of the heuristics module 520 detects an "occurrence of an abnormally high frequency of requests from different client devices 122 to execute the same software within a relatively short time period," which "is indicative of a software worm trying to spread among the client devices 122 and thus suggests that the software is malicious," (*see id.*, [0050]; *see also id.*,

[0068], [0031], [0011]), the execution authority classifies the software as malware. Namely, the execution authority maintains a database of software signatures and associated "status information indicating whether the software is malicious or benign." *Id.*, [0031], [0044], Fig. 5. Thus, classifying an application in an appropriate category indicated by the application-related data without the aid of a human reviewer was within the level of a person having ordinary skill in the art at the time of the invention.

107. For these reasons, a skilled artisan would have understood that *Kester* or *Kester* in combination with *Honig* teaches *classifying the computer object as malware when the data indicates that the computer object is malware*.

(e) *[1.4] when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware;*

108. In my opinion, *Kester* teaches classifying the program/application as something other than malware when the application-related data indicates that the program/application is not malware—such as Access/Privacy—and therefore teaches Element 1.4.

109. Regarding classification performed at the application server (*see Kester*, [0120] and Fig. 10), a reviewer can assign the application into one of the several categories or parent groups discussed above in Figure 4A. These "exemplary categories of applications include operating systems, anti-virus software, contact

managers, collaboration, media players, adult, and malicious applets and scripts. The categories can be further grouped into parent groups." *See id.*, [0082]. For example, parent groups might include "system, access/privacy, productivity, communication, audio/video, entertainment, and malware." *See id*. at [0082], FIG. 4A.

110. Where classification is performed at the application database factory, *Kester* teaches classifying the application into one or more "appropriate" categories (*see Kester*, [0092]), including these same categories or parent groups. In one example, the application-related data for the "CMD PCI IDE Bus Driver" application in Figure 6 includes the publisher (CMD Technology, Inc.), related suite (Microsoft Windows Operating System), filename, and directory location, indicating that the application is not malware. *Id.*, [0097]; Fig. 6. Accordingly, "the application analyst's classification module 302 classified the application in the parent group titled "access/privacy" and "operating system" category. *Id*., [0098], Fig. 6; *see also Kester*, Claim 19. This is shown below in Figure 6:

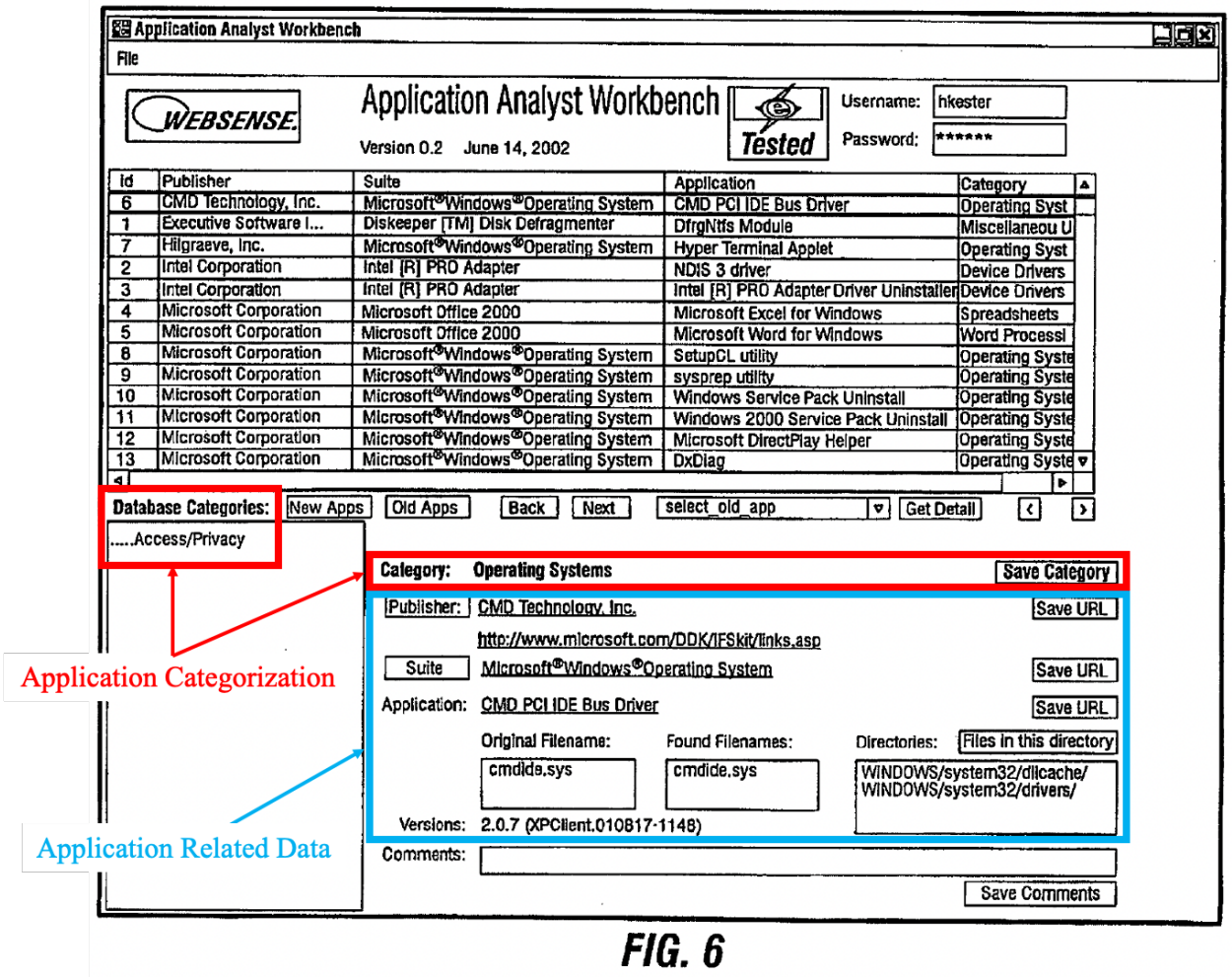


FIG. 6

Kester, Fig. 6 (annotated). A skilled artisan would have understood that categorizing an application into the parent group "access/privacy" and a category of "operating system" includes categorizing the application as something other than malware. Moreover, when *Kester* teaches when the application related data indicates an application is not malware, the application is assigned (i.e., *classified*) into the respective appropriate non-malware category or categories.

111. For these applications which are categorized as something other than malware, "the execution launch detection module 210 initially evaluates a launched

application and allows the application to run on the workstation based on the policy associated with the category or group of the application." *Kester*, [0137]. The operation of the application continues to be monitored by the network access detection module so that "even though an application is allowed to launch on a given workstation, the network access detection module 208 may curtail or limit the application if the application does not operate in a predetermined manner." *Id.*, [0138]. "A categorized application that does not operate in a predetermined manner is placed in the uncategorized applications database 108," which will cause the application to be recategorized, possibly as malware as discussed above. *Id*., [0070].

112. As I discuss in more detail in Element 1.10 below, *Kester* teaches "[a] human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization," meaning an application can be initially classified into one category such as the operating systems category of the access/privacy parent group and subsequently recategorized into a different category such as malicious scripts or malware. *Kester*, [0089].

Alternative Combined Teachings of Kester and Honig

113. Alternatively, it is my opinion that *Kester* in view of the teachings in *Honig* also renders this element obvious.

114. In my opinion, it would have been obvious to implement *Kester's* application classification process, such that the application server module and/or

database factory initially classify and/or reclassify applications into a parent group/category—including non-malware groups and categories—without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above.

(f) *[1.5] automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data;*

115. As I explained above in ¶ 56, this limitation recites "automatically," and pursuant to the petition's proposal, I have assumed that human action is excluded and that the automatic processes must be performed entirely without human involvement. I take no position as to the correct interpretation of the claims with respect to human involvement.

116. Furthermore, I have been informed that Patent Owner interprets the plain meaning of these "mask" limitations to "encompass[] any suitable way to define acceptable behavior" for an object. I have been asked to apply this meaning to these "mask" limitations.

117. *Kester* teaches that its application categorization and classification process involves determining an application's "**expected or allowed network activity**" (i.e., *generating a mask*). *Kester*, [0179]-[0181], Fig. 20. More specifically, *Kester* derives this expected network activity for an application "from network

access data obtained when the application is behaving in a predetermined manner." *See Kester* at [0070], [0046]. *Kester* teaches that a human reviewer (at the application server module 102 and/or application database factory 110) interacts with a classification user interface to determine the "expected or allowed network activity" (also referred to as expected or predetermined behavior) for an application from, for example, a record of the "application's prior activity" on one or multiple workstations:

> Next, at a state 2004, the application analyst's classification module 302 is utilized to display the application and any related data on the GUI. The related data can indicate to the human reviewer the expected network activity for the application. As explained above, the application analyst classification module 302 allows the human reviewer to analyze each application and any additional data that is associated with the application to determine an expected or allowed network activity. …
>
> The expected or allowed network activity can be based upon prior or contemporaneous network activity for the same application. For example, the expected network activity for an application running on a first workstation 101 can be determined from a record of that application's prior activity on the first workstation. In addition or in the alternative, the expected network activity for an application is determined from a record of that application's prior activity on multiple workstations.

Id., [0179]-[0181]; *Kester*, [0183] ("the network activity from multiple workstations is uploaded to the application database factory 110"); *see generally Kester*, [0178]-[0182], [0043]-[0044], [0069]-[0070], [0163]-[0164], Figs. 17, 20.

118. Once determined, the expected or allowed network activity of the application is stored in the hash/policy table by the workstation management module (*see id.*, [0183], [0045]; *see also id.*, [0053]-[0054], [0077], [0147]-[0151], [0107], Fig. 15), in the master application database of the application database factory (*see id.*, [0187], Fig. 20 (step 2010)), and in the application inventory database of the application server module (*see id.*, [0068]; *see also id.*, [0172]-[0174], [0084], [0107], Fig. 19).

119. *Kester's* expected network activity includes one or more network attributes, also referred to as properties or features, that are associated with the application when the application accesses the network in an expected (*normal*) manner. The attributes can include a specific protocol, IP address and/or port. *Kester*, [0046], [0184]; *see also Kester*, [0070] ("expected features relate to normal network activity by the application"); *Kester*, [0083] ("The network access data [including protocol, IP address, and/or port] for the application allows the network administrator to discriminate between expected/predetermined behavior and unexpected behavior of the application."); *Kester*, [0083], Fig. 4B.

120. A skilled artisan would have understood that determining the "expected network activity" based on prior or contemporaneous network activity, as taught by *Kester*, teaches *generating a mask for the program/application that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the application/program determined from said received data*. Like *Kester*, the '250 Patent discloses a mask that "can be built up with use of that object," "can give an overall picture of the expected 'normal' behaviour of that object," and might indicate "details of any ports or interfaces" for legitimate Internet connections. *'250 Patent,* 14:55-58.

Combined Teachings of Kester and Honig

121. *Kester* explains, however, that its "expected or allowed network activity" is determined by a human reviewer. *See Kester*, [0083], [0179], [0183], [0187]). Based on my review of the file history, I understand that the application argued during original prosecution that the term "**automatically**" in this element means a human decision is not necessary to generate the mask. *'250 File History*, 143.

122. However, *Kester* explains that the process for classifying or categorizing applications, such as to obtain the "expected or allowed network activity" (i.e., *mask*) as set forth in *Kester* at Figure 20, "can be based upon [] adaptive learning systems." *Kester,* [0089], [0092], Fig. 20. One of skill in the art,

therefore, would have found it obvious to look to known adaptive learning systems to perform *Kester's* classification and/or categorization processes. As I described in my opinions above, *Honig* teaches a known process for generating adaptive learning models to inform a malware determination/classification. *See* Elements 1.2 and 1.3.

123. Specifically, *Honig* identifies a need in the art to "automate the processes of data collection, model generation and data analysis" (*see Honig*, 4:53-57) "to look for clues of malicious behavior." *Id*., 1:54-57. As I identified above, *Honig* satisfies this need with an "Adaptive Model Generation" or AMG architecture, which "provides and automates many of the critical processes in the deployment and operation of real time data mining-based intrusion detection systems" (*Honig*, 6:50-54) including "[a]utomated data collection… used for training detection models… [and] to support various types of data analysis such as forensic analysis. Automated model generation trains detection models from data stored in the data warehouse. The process for converting the data into the appropriate form for training is fully automated." *Id.*, 7:9-20. *See above*, Element 1.2 (discussing types of models generated).

124. Like *Kester*, *Honig* teaches generating a mask of expected/allowed behavior based on normal activity. *Honig* accomplishes this using adaptive learning systems discussed above to automatically generate a model of behavior for a computer object in accordance with normal behavior without human input. *Honig*,

24:43-45 ("The model generator 16 then uses the unsupervised SVM algorithm to generate a model of normal activity"); *see also id*., 4:54-57 ("What is needed is an architecture to automate the processes of data collection, model generation and data analysis, and to solve many of the practical problems associated with the deployment of data mining-based IDSs."). In *Honig*, the model generation algorithms are "**fully automated**." *Honig*, 7:17-20; *see also id.*, 20:33-21:38. This model of normal behavior can then be used to make a malware determination/classification.

125. It would have been obvious to a skilled artisan to "automatically" generate (i.e., without human input) *Kester's* mask based on the teachings of *Honig*. More particularly, a skilled artisan would have been motivated to generate *Kester's* "expected or allowed network activity" (i.e., *mask*) for an application using an adaptive learning model as disclosed in *Honig* because it is expressly contemplated by *Kester. Kester*, [0089], [0092], Fig. 20 [0090]. A skilled artisan would have been familiar with adaptive learning systems as of 2005 in the context of intrusion and malware detection systems and would have understood that such systems require no human input. A skilled artisan considering the teaching of *Kester* that generation of the mask can be based upon "adaptive learning systems" would have found it obvious to generate the mask automatically (e.g., without any human decisions) with any of the well-known and readily available adaptive learning systems—such as *Honig's* readily available adaptive learning system.

126. A skill artisan would have also recognized that *Honig* provides an express benefit for implementing the combined method/system in this way—to allow the method/system to "update and alter models on the fly[,]" consistent with *Kester's* framework for allowing an administrator to adjust/control prohibited behaviors. *Honig*, 19:46-62. Indeed, a skill artisan would have recognized implementation of automated mask generation to be an obvious and predictable improvement to *Kester's* server module and/or database factory classification software, given that *Kester* expressly contemplates use of adaptive learning systems.

127. Furthermore, *Honig* recognized the need for an extensible and scalable system where the model generator components are modular components that are "plugged" into the system architecture, which would easily be compatible with *Kester's* process. *Honig*, 7:38-41 ("The system is designed to be extensible and scalable, and hence any audit source, any supervised learning module, any unsupervised (anomaly detection) learning model may be easily inserted into the architecture."); *see also id*., 13:61-66.

128. Automating generation of a mask of expected/allowed behavior, as taught by *Honig*, would have made *Kester's* method/system more desirable and more efficient for a human reviewer to use, thereby improving similar malware detection method/systems in the same way. *Kester*, [0090], [0135] ("hints" enhance productivity of human reviewer for classification). Indeed, *Honig* notes that human

intervention, such as manually labeling or cleaning training data, is expensive and time consuming. *Honig*, 2:40-42, 3:2-4. *Honig* states that automated models allow the system "to update and alter models on the fly with minimal computational overhead. This is very advantageous because it allows the system to be deployed for a long period of time without the need for maintenance by an administrator." *Id*., 19:46-62. Thus, by automatically generating the masks defining expected behavior of *Kester's* applications using the SVM algorithms of *Honig*, a person having ordinary skill in the art would have appreciated the reduced costs of implementation, both in "the expense in time and human expertise" of manually performing tasks that could be automated. *Id.*, 2:2-4; 3:24-30.

129. Additionally, *Honig* also provides a detailed discussion of adaptive learning model generators that were known in the field as of 2005 and provides a detailed discussion of example implementations. *Honig*, 2:5-4:47, 21:39-24:31. Notably, *Honig* references a technique similar to adaptive model generation developed at SRI in the Emerald system in 1993, which uses historical records to build normal detections models and compares distributions of new instances to historical distributions to identify discrepancies signifying intrusions. *See id.*, 3:31-41; *Javitz et al, "The NIDES Statistical Component: Description and Justification"* (Ex. 1019).

130. Another known example of automatically generating a mask is described in U.S. Patent No. 7,516476 to Kraemer ("*Kraemer*"). *Kraemer* teaches "automatically creating a security policy for one or more applications[.]"

> "As the applications execute, the access requests they issue are monitored and recorded. This data is aggregated into a set of 'representative' actions for the application(s) according to a heuristic which… is configurable by automated or manual means. The representative actions provide input to a process which develops security policy for the application(s) according to a template. The template may define the structure of the security policy based, at least in part, on the previous behavior of the one or more applications on similar systems."

See Kraemer, 4:16-29. The "automated creation of a security policy" includes "gathering behavioral data from the application(s) for which the security policy is to be developed," monitoring "requests issued by the application(s) as features are exercised," and analyzing "the data collected" to produce a security policy. *Id*., 5:11-16. *Kraemer* states that the applications "may execute on one or more workstations on the system[.]" *Id.*, 5:35-36. "Policy generation module 132 analyzes the data to determine how the application(s) issue requests to access resources, the types of requests made, requests typical to certain system resources, and other observations." *Id.* 7:4-7. Importantly, Kraemer explains that the "process of developing security policy… may be performed in a semi-automated fashion (e.g., wherein a human

operator intervenes to initiate analysis job 116) or completely automated fashion, **wherein no human intervention is required**." *Id*., 11:49-58.

131. Accordingly, modifying a mask automatically (i.e., without human input) was well-known in the art by 2005. Thus, modifying *Kester* to implement an automatically generated behavior mask, as taught by *Honig*, would have been well within the capabilities of one of ordinary skill in the art such that a skilled artisan would have had a reasonable expectation of success.

132. Thus, generating *Kester's* expected/allowed application behavior automatically, as disclosed by *Honig*, renders obvious *automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received*.

(g) *[1.6] running said object on at least one of the remote computers;*

133. *Kester* discloses running the same program/application (*said object*) on one or more remote workstations.

> [E]ach time an application that the workstation management module 200 has allowed to run on the workstation 101 attempts to accesses a network, the workstation management module 200 determines whether to allow the application to access the network. In this way, the workstation management module 200 can keep or change a previous access privilege based on the subsequent activity of the application.

Kester, [0041]; *Kester*, [0157] ("different workstations 101 can have different policies associated with the same application running thereon"); *Kester*, [0116] ("the same application may be allowed to run on a workstation but not allowed to run on a different workstation"); *see generally Kester*, [0137], [0049]-[0055], [0057], [0059]-[0060], [0099]-[0104], Abstract, Figs. 2, 7; *see also* element 1.7.

134. Thus, in my opinion *Kester* teaches *running said object on at least one of the remote computers*.

(h) *[1.7] automatically monitoring operation of the object on the at least one of the remote computers;*

135. *Kester* discloses a network access detection module 208 on each remote workstation that continually and automatically monitors the operation of the program/application (*the object*). *See, e.g.*, *Kester*, [0137]-[0138]:

> [T]he execution launch detection module 210 initially evaluates a launched application and allows the application to run[.] …
> [T]he subsequent operation of the application is monitored by the network access detection module 208. Thus, even though an application is allowed to launch on a given workstation, the network access detection module 208 may curtail or limit the application if the application does not operate in a predetermined manner. Further, the network access detection module 208 can continually or periodically monitor the running application to ensure that the applications continues to operate in the predetermined manner.

See also Kester, [0041] ("[T]he workstation management module 200 can monitor the ongoing network activity or behavior of the application…For example, each time an application that the workstation management module 200 has allowed to run on the workstation 101 attempts to accesses a network, the workstation management module 200 determines whether to allow the application to access the network."), [0051] ("When an application or program accesses a network, the network access detection module 108 monitors the behavior or activity of the application or program."); [0139] ("FIG. 14 is a flow diagram illustrating a process for monitoring the behavior of an application."); *see generally id*., [0137]-[0146], [0039], [0050]-[0055], [0057]-[0060], Figs. 2, 14.

136. Because *Kester* discloses that the "network access detection module 208" performs the above-described monitoring operations upon launching the application, when the application attempts to access the network, continually, and otherwise periodically without human involvement, a person having ordinary skill in the art would have understood this monitoring to occur "automatically."

137. Therefore, *Kester* teaches *automatically monitoring operation of the object on the at least one of the remote computers*.

(i) *[1.8] allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;*

138. *Kester* teaches this element. For example, *Kester* discloses:

> [T]he execution launch detection module 210 initially evaluates a launched application and allows the application to run on the workstation based on the policy associated with the category or group of the application… However, the subsequent operation of the application is monitored by the network access detection module 208. Thus, even though an application is allowed to launch on a given workstation, the network access detection module 208 may curtail or limit the application if the application does not operate in a predetermined manner. Further, the network access detection module 208 can continually or periodically monitor the running application to ensure that the applications continues to operate in the predetermined manner.

Kester, [0137]-[0138]. In other words, *Kester* allows an application to run on a workstation and thereafter monitors the application's behavior—when the running application "continues to operate in a predetermined manner" (*behavior of the computer object is permitted by the mask*), the application is allowed to continue running. The condition under which the network access detection module "may curtail or limit the application" is "if the application does not operate in a predetermined manner" (behavior of the computer object is not permitted by the mask, discussed in Element 1.9 below). *See also id*., [0051], [0055], [0057], [0143].

139. *Kester's* Figure 14 clearly illustrates the process of allowing an application to run based on its predetermined behavior. First, the application digest generator 201 "generates a digest of data relating to the application" including

properties/attributes such as IP address(es), port(s), protocol, and/or other network access data (also referred to as "collection data"). *Kester*, [0141]; Fig. 14 (step 1404). Second, the network access detection module 208 of the workstation 101 "compares the application digest and collection data prepared by the application digest generator 201 to the hash/policy table 204." *Kester*, [0142]; Fig. 14 (step 1406). Third, when "the behavior or network attributes of the application matches with an expected behavior for the application", the associated one or more access privileges/policies/rules, such as "allowing execution of the program" and/or "allowing the application to access the network" are applied. *Kester*, [0143]; Fig. 14 (step 1410). *See generally Kester*, [0050]-[0057], [0137]-[0145]. This Figure 14 process is illustrated below:

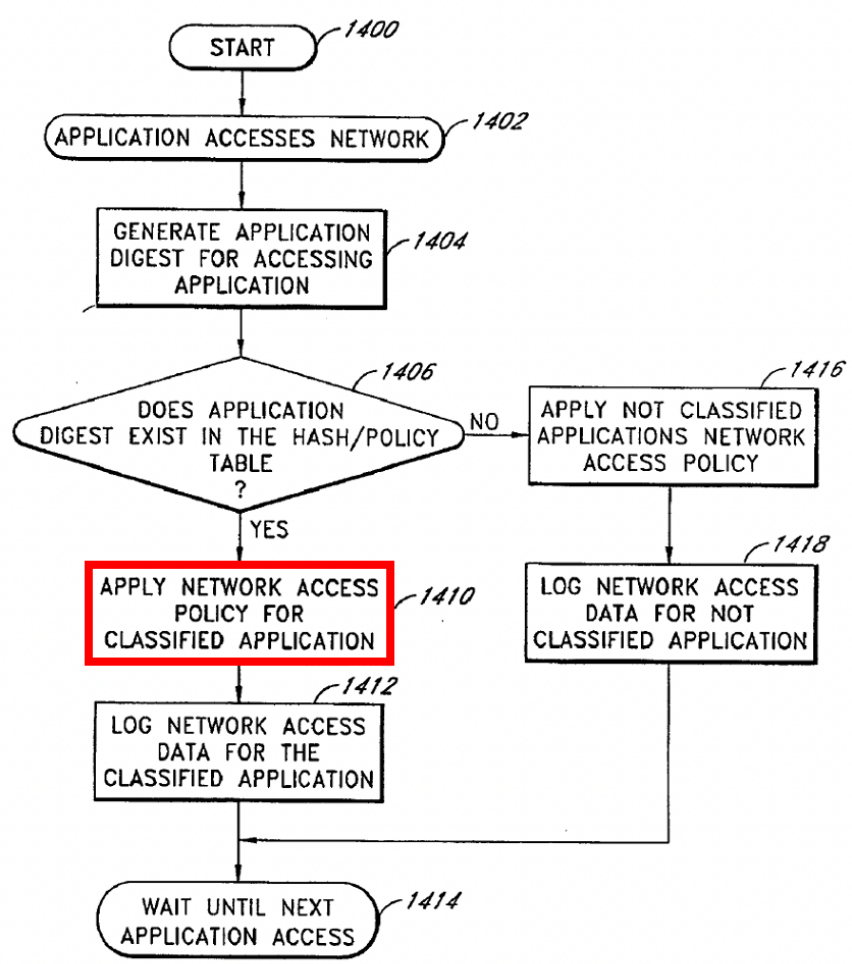


FIG. 14

Kester, Fig. 14 (annotated)

140. Thus, a skilled artisan would have understood that *Kester* teaches *allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask*.

(j) *[1.9] disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and,*

141. In addition to the above in Element 1.8, when an application is not operating in a predetermined manner, this application is curtailed or limited, such that it is denied access to the network. *Kester* teaches "denying execution of the program" and/or "denying network access to the network" (*disallowing the computer object to run*) when the network accessing program/application properties/attributes (t*he actual monitored behavior of the computer object*) do not match with an expected behavior for the application (*extends beyond that permitted by the mask*). *Kester*, [0055], [0059], [0145]. In particular, when an application is behaving unexpectedly, the "execution launch detection module 210 or the network access detection module 208 can stop execution or network access for the application[.]" *Kester*, [0059]. *Kester* also describes "denying execution of the program" and/or "denying network access to the network" when the program/application behavior does not match with an expected behavior for the application. *Kester*, [0055], [0059], [0145].

142. For this denial process, the expected network activity stored in the hash/policy table at each remote workstation includes "attributes [that] are associated with the application when the application accesses the network in an expected manner" (e.g., a specific protocol, a specific IP address, and a specific access port). "If the application requests access to the network using a different protocol than the expected protocol listed in the hash/policy table 204, the network

access detection module 208 may disallow access." *Kester*, [0184]. The process of disallowing a computer object to run when the actual monitored behavior extends beyond that permitted by the mask includes first generating application digest and second comparing digest and collection data to hash/policy table 204. *See* above, Element 1.8 for detailed discussion.

143. Third, when "the application digest and collection data does not correspond with an application or hash classified in the hash/policy table 204," this means that "the application is behaving unexpectedly." *Kester*, [0145], [0059]. Here, "the network access detection module 208 applies a not-classified application policy to the request to access to the network." *Kester*, [0145]; Fig. 14 (step 1410). Per *Kester*, the not-classified application policy/rule can include "stop[ping] execution or network access for the application," "denying execution of the program," and/or "denying access to the network." *Kester*, [0055], [0059], [0145]. *See generally Kester*, [0050]-[0059], [0137]-[0145], [0184]. This process is also shown in Figure 14:

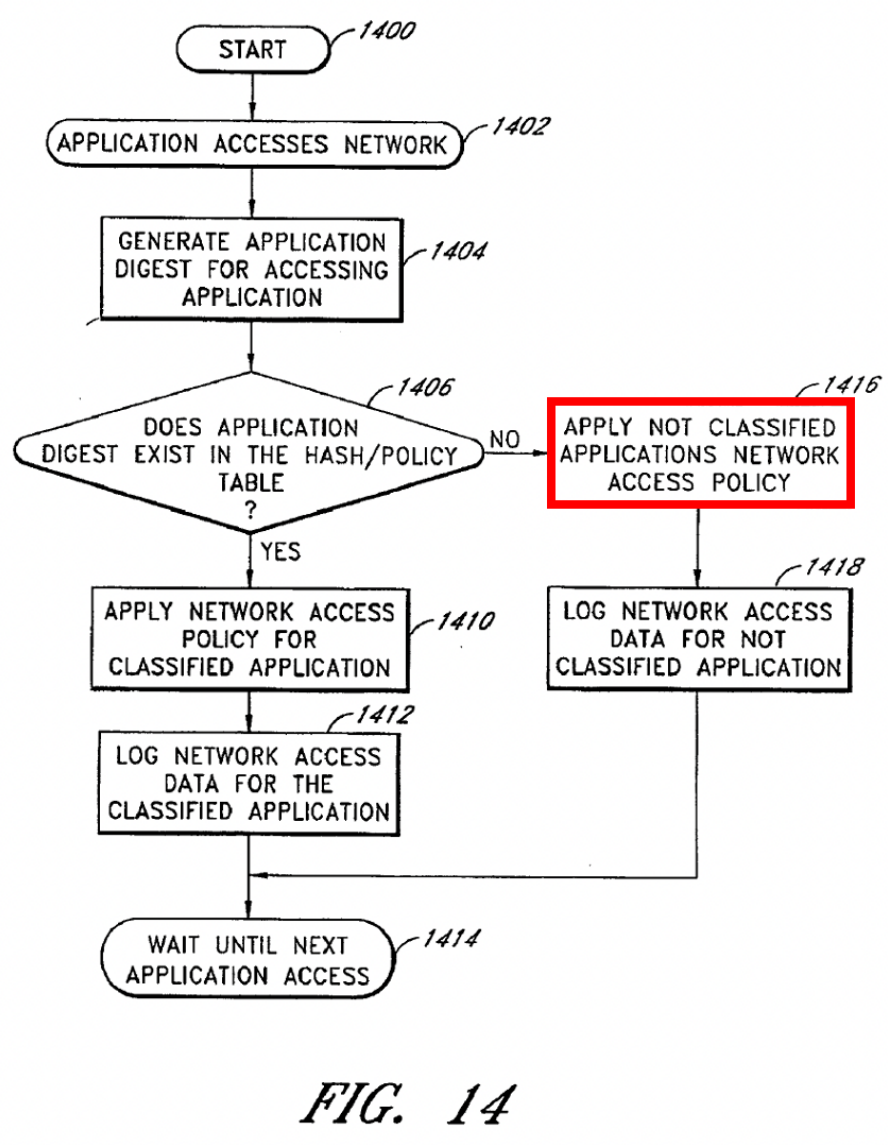


FIG. 14

Kester, Fig. 14 (annotated)

144. A skilled artisan therefore would have understood *Kester* discloses *disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask.*

(k) *[1.10] reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.*

145. In my opinion, *Kester* or *Kester* in view of *Honig* teaches re-categorizing (reclassifying) the program/application as malware when the program/application is not operating in an expected manner as set forth in my opinions below.

146. *Kester* teaches a process where an application that is not operating pursuant to its mask (i.e., not in a predetermined manner) is "categorized or re-categorized[.]" *Kester*, [0076]. Here, applications that are not operating in a predetermined manner, are "stored in the uncategorized application database 108." *See Kester* at [0075]. "The application is re-categorized based on parsed properties or features which are different than the parsed properties or features associated with the original categorization of the application." *Kester*, [0076]. Re-categorization can occur either by the "network administrator, via the classification user interface 106" or at application factory database 110. *Id.*

147. With respect to reclassification at the application factory database 110, an uncategorized application's application related data may be uploaded to the application database factory for reclassification/re-categorization, for example, "immediate[ly]" (e.g., upon receipt from the workstation(s)) (*see Kester*, [0078]) or when the human reviewer "does not classify the application" (*see Kester*, [0076]). *See also id*., [0078], [0084], [0122], Fig. 3. Here, at the application database factory 110, "[t]he application analyst classification module 302 can use SQL queries, in

conjunction with the rules, to select the subset [of applications] for categorization ***or recategorization*** from the master application database 300." *Kester*, [0091]. The reclassification/recategorization process in the application database factory is the same as the classification/categorization process discussed in Elements 1.2 and 1.3. *Kester*, [0089]-[0094].

148. Thus, in my opinion, *Kester* teaches reclassifying the program/application as malware when the application related data so indicates.

Alternative Combined Teachings of Kester and Honig

149. It would have been obvious to a skilled artisan to implement *Kester's* application classification process, such that the application server module and/or database factory initially classify and/or reclassify applications into the malware parent group/category without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above. *See* Element 1.2.

2. *Claim 2 – A method according to claim 1, wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.*

150. 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 "wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of" followed by a list of items. To render this limitation unpatentable, the prior art need only disclose one of the items in the above list.

151. As I described in my opinions above, *Kester* discloses that application related data (*data about the computer object*) is sent from multiple remote workstations (*plural remote computers*) to the application server module and/or database factory (*base computer*). *See above*, Element 1.1. Kester explains that this application related data may include "application name" (*current object name*), "original Filename" (*original object name*), "directory location" (*physical and folder location on disk*), "file size" (*size of the object*), "publisher, suite, … version" (*vendor, product and version*), and network access request "time of day, port, I.P. address, protocol" (*events initiated by or involving the object when the object runs on the respective remote computers*). *Kester*, [0061]-[0062], [0051], [0119], [0097], Fig. 6; *see also* Element 1.1. Additionally, the application related data sent from the remote workstations to the base station may include a hash generated from executable instructions contained within or constituted by the object (*executable*

instructions contained within or constituted by the object). *See below,* opinions concerning Claim 8.

3. *Claim 7 – A method according to claim 1, wherein the data is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on the respective remote computers.*

152. *Kester* explains that the application related data (*see* element 1.1 and claim 2) may be sent in the form of a hash (i.e., *key*) that is generated by each remote workstation using a hash function. *Kester*, [0051] ("As part of its analysis, the execution launch detection module 210 can generate a hash for the application using the application digest generator 201. The application digest generator 201 parses properties from the requested application to uniquely identify the application. These properties can include net-work access data. Exemplary properties include the name, publisher, suite, hash, file size, version, protocol, I.P. address, port, and additional information or properties which are associated with a launched or network accessing application").

153. *Kester* discloses that the "hash for the application is determined by transforming the binary associated with the application into a unique set of bits." *Id*., [0052]. "A hash function, which is a form of encryption known in the art, is employed in determining the hash for the application." *Id*. In this way, "the hash function takes selected binary input from the application and transforms the binary into a fixed-length encrypted output called a hash." *Id*. "The result is a hash with a

fixed-size set of bits that serves as a unique 'digital fingerprint' for the application." *Id*. Kester provides two "exemplary hash algorithms include[ing] MD-5 and Secure Hash Algorithm-1 (SHA-1)." *Id*. (noting "[t]he MD-5 hash algorithm produces a 128-bit output hash. The SHA-1 algorithm produces a 160-bit output hash"). Importantly, in embodiments, the "the hash value[] includes network access data." *Kester*, [0011], Claim 39.

154. These hashing processes described in *Kester* to generate a hash (i.e., *key*) were well-known in the art by 2005. For example, *McCorkendale* discussed above generally describes that software is identified by a signature that is generated using "the software as the input to the hash function" to obtain a hash. *McCorkendale*, [0040]. "The hash function is selected so that any change to the software will produce a change in the hash. Therefore, the hash acts as a sort of fingerprint of the software." *Id.* Example hash functions disclosed by *McCorkendale* include MD5 and SHA. *Id.*

4. *Claim 8 – A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.*

155. As I explained in my opinions above in claim 7, *Kester* describes that its "hash for the application is determined by transforming the binary associated with the application into a unique set of bits." *Kester*, [0052]. "[T]he hash function takes selected binary input from the application and transforms the binary into a fixed-

length encrypted output called a hash. The result is a hash with a fixed-size set of bits that serves as a unique 'digital fingerprint' for the application." *Id*. Example hash functions disclosed by *Kester* include MD-5 and SHA-1. *Id*.

156. Based on the above disclosure, a skilled artisan reading *Kester* would have readily understood *Kester's* hash (i.e., *key*) has at least one component that represents executable instructions contained within the object (e.g., object code, machine code, or other code readable by a computer when loaded into memory and used to execute instructions). I note that this is consistent with the '250 Patent's disclosure of an MD5 Type 1 checksum (i.e., *key*). *'250 Patent*, 2:13-17, 9:63-10:32; *see also* claim 7.

5. *Claim 9 – A method according to claim 7, wherein the key has at least one component that represents data about said object.*

157. As I described in my opinions above, *Kester's* hash (i.e., *key*) includes several types of components representing data about the application. *See* Claim 2. For example, *Kester's* hash can include network access data (*see Kester*, Claim 39, [0011]. "[N]etwork access data includes parsed properties or attributes that are associated with the application when the application accesses the network." *Kester*, [0069]. This includes, for example, the "time of day, port, I.P. address, protocol" of a network access request, which are events initiated by or involving the object when the object runs on the respective remote computers. *Kester*, [0061], [0051], [0011],

[0083]. Moreover, network access data includes the "name, publisher, suite, [] file size, [and/or] version" of the network accessing application. *Kester*, [0051].

158. Thus, in my opinion, *Kester's* hash value has *at least one component that represents data about the object*.

6. *Claim 10 – A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.*

159. *See* Claim 9.

7. *Claim 11 – A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.*

160. As I described above in my opinions concerning claims 2 and 7, *Kester's* hash (i.e., *key*) includes network access data including the network accessing application's "file size." *See*, Claims 2 and 7; *see also Kester*, [0051]; *id*., [0011], Claim 39.

161. Thus, it is my opinion that *Kester's* hash value has *at least one component that represents the physical size of the object*.

8. *Claim 12 – A method according to claim 1, comprising if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the*

non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future.

162. As I explained above, I have been informed that Patent Owner has interpreted the claims in litigation, including this claim, claim 12, to encompass human involvement. *See above*, ¶ 56. I have been asked to apply Patent Owner's interpretation to claim 12. I take no position as to the correct interpretation of the claims with respect to human involvement.

163. *Kester* explains that when an application that is behaving unexpectedly (*the monitored behavior exhibits non-permitted behavior that extends beyond that permitted by the mask*), it is considered an "uncategorized application" and its application related data (including the unexpected behavior) is transmitted to the server module and/or database factory (*base station*) for categorization/re-categorization. *Kester*, [0070]-[0071], [0075]-[0076], [0059], [0084], [0087], [0151]-[0153]; *see above* Element 1.10.

164. *Kester* further explains that an application that is "not operating in a predetermined manner [is] categorized or re-categorized" and that "[t]he application is re-categorized based on parsed properties or features which are different than the parsed properties or features associated with the original categorization of the application." *Kester*, [0076]. Thus, in my opinion, *Kester* teaches comparing at the base computer the non-permitted behaviour (e.g., "the parsed properties or features

that are different") with the behaviour of other objects (e.g., "the parsed properties or features associated with the original categorization of the application").

165. In the scenario where an application is behaving unexpectedly, "the network access detection module 208 [of the remote workstation] applies a not-classified application policy to the request to access to the network." *Kester*, [0145], [0059], Fig. 14 (step 1410). In this instance, in *Kester*, the not-classified application policy/rule can include "allowing execution of the program" and/or "allowing the application to access the network." *Kester*, [0055], [0059], [0145]; *see generally Kester*, [0050]-[0059], [0137]-[0145], [0184], Fig. 14.

166. *Kester* provides that categorization and re-categorization can include categorizing/re-categorizing the expected or allowed network activity for the application. *See, e.g., Kester*, [0089] ("A human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization. The process for classifying or categorizing applications at the application database factory is described with reference to FIGS. 13 and 20."), *Kester*, [0178]-[0187] and Fig. 20 (step 2002, where the list of uncategorized applications is extracted for classification and step 2008, "determine allowed behavior for each application"); *see also id.*, [0071], Fig. 17. I understand categorizing or recategorizing the unexpected or allowed network activity for the application to include "*modifying the mask*" as claimed. For instance, one of skill in

the art would have understood *Kester* to disclose re-categorizing the expected or allowed network activity for an application and that such re-categorization may be based on parsed properties or features which are different than the parsed properties or features associated with the original categorization of the application.

167. In the context of categorizing and/or re-categorizing, *Kester* explains that a human reviewer may determine whether behavior is allowed, including by analyzing network activity from multiple other workstations and performing additional research. *See, e.g., Kester*, [0178]-[0187], [0163]-[0164], Figs. 17, 20. A skilled artisan would have understood *Kester's* disclosures here to encompass a situation where the unexpected behavior is known to be not malicious for the same application operating on other workstations to which it is compared and re-categorizing (i.e., *modifying*) the expected or allowed network activity for the application (*mask*) to allow the behavior for the application in the future.

168. Additionally or alternatively, *Kester* in view of *Honig* discloses this process. *Honig* teaches a model generator 16 that automatically generates a model of behavior for a computer object in accordance with normal behavior. *See, e.g., Honig*, 8:15-18, 24:43-45, 4:54-57; *see also* Claim 1, Element 1.5. *Honig* further teaches that models "become out of the data" and are "update[ed] and alter[ed] on the fly." *Honig*, 19:46-62. Updated models are automatically distributed so that detectors have the "most recent" model for properly classifying behavior as normal

or malicious. *Honig*, 19:46-62, 14:42-51, 14:28-31, 8:14-24. Moreover, *Honig* incorporates by reference *Apap* (U.S. application Ser. No. 10/352,343, now U.S. Patent No. 7,448,084 to Apap et al. ("*Apap*")) (Ex. 1020) as disclosing an intrusion detection system that is integrated into *Honig's* system and uses an anomaly detection algorithm to make models of normal registry accesses. *Honig,* 24:50-57, 11:60-67, 25:44-50. *Apap* discloses that unexpected behavior may actually represent normal operation. *Apap,* 7:47-59; *see also id*., 8:1-7 ("[T]he user is prompted when a detection occurs, which provides the user with the option of informing the algorithm that the detected program is not malicious and therefore grants permission for such program to be added to the training set of data to update the anomaly detector for permissible software").

169. Indeed, this concept was known in the art. For example, *Kraemer* teaches that "[t]he system may also be configured to alert a user when an application performs an action which is not prohibited per se by a security policy, but which the system deems abnormal." *Kraemer*, 11:31-33, 11:36-42 ("Once the user has been alerted, the user may request that the security policy adapt to prevent the considered action from subsequently occurring [and] the user may decide to modify the security policy or leave it as is").

170. It would have been obvious to a POSITA to configure *Kester's* server module and/or database factory user interface to alert the human reviewer when

unexpected behavior extends occurs and allow for the user to indicate that the unexpected behavior is known to be not malicious and to modify the mask to allow the behavior for the application in the future based on known in the art teachings, such as those in Honig. A skilled artisan would have recognized this implementation to be an obvious and predictable improvement to *Kester's* application file monitoring system/method. As noted above, *Kester's* express disclosures already encompass this type of implementation. In these implementations, a skilled artisan would have been motivated to apply the teachings of *Honig* to ensure that when unexpected behavior actually represents normal operation, *Kester's* expected network activity mask is modified accordingly. This would have several benefits, including reducing false alarms and nuisance to *Kester's* employee end users. Thus, this implementation would have made *Kester's* method/system more desirable and more effective in identifying malicious/rogue programs (malware), thereby improving similar malware detection method/systems in the same way.

171. A skilled artisan would also have understood that this combination of prior art elements would have been accomplished using known methods and with no change to their respective functions. For instance, *Kester's* application file monitoring system/method would still allow for human research and analysis of the application's activity on other workstations when re-categorizing expected network activity of an application while also implementing *Honig's* known method of

alerting a user when unexpected behavior occurs and allowing the user to indicate that the unexpected behavior is not malicious and that the mask should be updated accordingly. Furthermore, given the triviality and prevalence of providing such alerts and prompting for user input, there would have been a reasonable expectation of success configuring the combined method/system in this way.

172. Thus, in my opinion, *Kester* or *Kester* in view of *Honig* renders Claim 12 obvious.

9. <u>*Claim 13*</u>

(a) *[13(preamble)] Apparatus for classifying a computer object as malware, the apparatus comprising:*

173. *See* Elements 1.2 and 1.3 above.

(b) *[13.1] a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;*

174. *See* Claim 1, Element 1.1.

(c) *[13.2] the base computer being constructed and arranged to determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,*

175. *See* Claim 1, Element 1.2.

(d) *[13.3] the base computer being constructed and arranged to classify the computer object as malware when the base computer determines that the data indicates that the computer object is malware,*

176. *See* Claim 1, Element 1.3.

(e) *[13.4] wherein the base computer is constructed and arranged to automatically generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data,*

177. *See* Claim 1, Elements 1.4 and 1.5.

(f) *[13.5] wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers,*

178. *See* Claim 1, Element 1.6 and 1.7.

(g) *[13.6] the object being allowed to continue to run when behaviour of the object is permitted by the mask,*

179. *See* Claim 1, Element 1.8.

(h) *[13.7] the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and*

180. *See* Claim 1, Element 1.9.

(i) *[13.8] the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.*

181. *See* Claim 1, Element 1.10.

10. *Claim 14 – Apparatus according to claim 13, the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header*

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

182. *See* Claim 2.

11. *Claim 19 – Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.*

183. *See* Claim 7.

12. *Claim 20 – Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.*

184. *See* Claim 8.

13. *Claim 21 – Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.*

185. *See* Claim 9.

14. *Claim 22 – Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.*

186. *See* Claim 10.

15. *Claim 23 – Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.*

187. *See* Claim 11.

16. *Claim 24 – A method according to claim 13, wherein, if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, the base computer is arranged to compare the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects the base computer is arranged to allow said behaviour for the object and to modify the mask to allow that behaviour for that object in future.*

188. *See* Claim 12.

17. *Claim 25*

(a) *[25(preamble)] A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:*

189. As I described in my opinions above, *Kester* discloses a method of providing data about a computer object from a remote computer to a base computer. *See* Claim 1, Element 1.1 above (describing remote workstations sending application related data to an application server module and/or application database factory). At the application server module 102 and/or application database factory 110, this applicated related data is compared with similar data received from other remote workstations. *Kester*, [0069] ("The network access data is uploaded to the application server module 102. The uploaded network access data can be **compared** to expected network access data for the application. As explained above, the expected network access data **can be a compilation of network access data**

associated with contemporaneous or prior network access by the application. The expected network access data can be compiled from the requesting workstation or from other workstations."), [0183] ("In a preferred embodiment, the network activity from multiple workstations is uploaded to the application database factory 110" and "[t]he network activity of the same application running on different workstations can be weighted in a predetermined manner to determine an expected network activity for the application."); *see also id*., [0076], [0179]-[0187], Figs. 10, 13, 17, 20. *See also* Claim 1, Element 1.5.

190. Therefore, in my opinion, *Kester* describes *a method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers*.

> (b) *[25.1] providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;*

191. *See* Claim 1, Element 1.1.

> (c) *[25.2] the data including information about the behaviour of the object running on one or more remote computers and including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote*

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

192. *See* Claim 1, Element 1.1 and Claim 2.

(d) *[25.3] the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer,*

193. *See* Claims 7-11.

(e) *[25.4] automatically generating a mask for the computer object, the mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data;*

194. *See* Claim 1, Element 1.5.

(f) *[25.5] executing the computer object at the remote computer;*

195. *See* Claim 1, Element 1.6.

(g) *[25.6] automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;*

196. As I described in claim 1 above, *Kester* teaches a network access detection module 208 on each remote workstation that continually and automatically monitors the operation of the program/application (*the object*). *See* Claim 1, Element 1.7. The network access detection module 208 in so doing, "a hash generated by the application digest generator 201 and collection data can be compared to hashes from the hash/policy table 204 and network access data associated with the hash" where the "network access data associated with the hash" is the "expected behavior for the

application" (*behavior permitted by the mask*). *Kester*, [0142]-[0143]; *see also id*., [0053], [0055], Fig. 14 (step 1406).

(h) *[25.7] allowing the object to continue to run when behaviour of the computer object is permitted by the mask;*

197. *See* Claim 1, Element 1.8.

(i) *[25.8] disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and,*

198. *See* Claim 1, Element 1.9.

(j) *[25.9] reclassifying the object as malware when the actual monitored behaviour extends beyond hat [sic] permitted by the mask,*

199. *See* Claim 1, Element 1.10.

(k) *[25.10] wherein the base computer is arranged to receive information about the behaviour of the object running on one or more remote computers,*

200. *See* Claim 1, Element 1.1.

(l) *[25.11] wherein the step of generating a mask for that object comprises building the mask with the base computer in accordance with normal behaviour of the object determined from said received information.*

201. *See* Claim 1, Element 1.5.

18. *Claim 26 – A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.*

202. *See* Claim 8.

19. *Claim 27 – A method according to claim 25, wherein the key has at least one component that represents data about said object.*

203. *See* Claim 9.

20. *Claim 28 – A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.*

204. *See* Claim 10.

21. *Claim 29 – A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.*

205. *See* Claim 11.

22. *Claim 30 – A non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.*

206. *Kester* discloses an "addressable storage medium" (*non-transitory storage medium*) storing software (*a computer program*) comprising program instructions for causing a computer to perform the method steps of claim 25. *Kester*, [0034] ("[M]any of the components of the various systems which may be included in the entire system, some of which are referred to as modules, can be implemented as software, firmware or a hardware component, such as a field programmable gate array (FPGA) or application specific integrated circuit (ASIC), which performs

certain tasks. Such components or modules may be advantageously configured to reside on the addressable storage medium and configured to execute on one or more processors."); *see Kester* applied to Claim 25. Where *Kester* is combined with *Honig* in the mapping of Claim 25 above, it would have been obvious to a POSITA to program *Kester's* software as discussed in Claim 25 at least because *Honig* similarly discloses computer-implemented methods. For example, *Honig* also teaches that its methods are computer implemented and provides a "computer listing" appendix of code routines for implementing the disclosed methods. *Honig* at 1:22-29.

H. Ground 2: *Kester* in view of *Honig* in further view of *Kennedy* Renders Claims 3-6 and 15-18 Obvious

1. *Claim 3 – A method according to claim 1, wherein determining identifies relationships between the object and other objects.*

207. As I discussed in my opinions above, *Kester* discloses that its application server module 102 and/or application database factory 110 (base computer) receives application related data, e.g., application name, file size, directory location, version, other network access data, execution request frequency, "and additional information associated with the requested application." *See above*, Claim 1, Elements 1.1-1.3. In addition, *Kester* teaches that "expected network activity can be determined from network activity **by a different but related application**." *Kester*, [0044], [0182]. Based on this disclosure, a skilled artisan would have understood that this process for determining "expected network activity"

requires identifying "different but related objects" in the base computer. However, *Kester* does not describe how it determines an application is "different but related." *Kennedy* describes such a process, as discussed below.

208. *Kennedy* discloses a malicious software detection module (MSDM) that "tracks the set of files that are associated with suspicious software." *Kennedy*, Abstract. Specifically, the MSDM includes a "file set tracking module 312" that "maintains a set of files for each suspicious software" and "tracks the behaviors of the files in the set and records each time a file in the set 'touches' [(e.g., creates, modifies, and/or reads)] another file on the storage device 108 or elsewhere in the computer system 100." *Id.*, 4:19-33. In this process, each touched file is added to the set of the file that touched it. *Id.*, 4:33-34. *Kennedy* explains that file sets and monitored behavior may be stored in a database. *Id.*, 4:48-57. Alternatively, "a file's attributes can indicate that it is a member of a particular file set and/or associated with particular suspicious software" and that the tracking module 312 can update the attribute data "to reflect activity by the suspicious software being monitored." *Id.*, 4:48-63.

209. *Kennedy* also describes a behavior monitoring module 316 that "monitors the behaviors of the files within the set associated with each suspicious software," and "employs one or more heuristics to determine whether the suspicious software represented by each set is malicious." *Kennedy*, 4:64-5:7. *Kennedy*

specifies that "[e]ach heuristic describes one or more conditions that, if satisfied, indicate that the software is malicious." *Id.*, 5:7-9. Such heuristics were well-known in the art by 2005, at least as exemplified by *McCorkendale* above. When tracking related files, *Kennedy* explains: "[a] heuristic is satisfied if any file in the software's set fulfills the conditions of the heuristic, and/or the collective actions of the files in the set fulfill the conditions of the heuristic." *Kennedy*, 5:9-11. *Kennedy* further explains that many different heuristics can be used to detect worms or malicious software from related file behavior, including heuristics that specify certain network activities (e.g., a file attempting to send a copy of itself and/or any other file in the same set). *Id.*, 5:12-33. Thus, *Kennedy's* determination that behavioral data of related files indicates that the software is a worm or otherwise malicious is a determination that the behavioral data indicates that the software is malware, consistent with the '250 Patent's definition of malware. *'250* Patent, 1:14-17.

210. It would have been obvious to a skilled artisan to configure *Kester's* system/method for monitoring application files such that each remote workstation additionally includes a file set tracking module as disclosed in *Kennedy*. Here, the workstations comprising a file set tracking module would log the set of files for each application in the logging database and would log each time a file in the set touches (e.g., creates, modifies, and/or reads) another file. It also would have been obvious to configure *Kester's* application server module and/or database factory to determine

whether received data indicates an application is malware based on an identified relationship between application file(s) and different but related files as taught by *Kennedy*. In the combined method/system, the application's file set behavior is application related data that is sent to the server module and/or database factory and used by *Kester* to determine whether the behavior is indicative of malware as taught by *Kennedy*. I provide a working example below based on *Kennedy's* express disclosures.

211. In one example, sus-a.exe directly or indirectly creates a new file, sus-b.exe, and attempts to send sus-b.exe to another computer system. *Kennedy*, 4:19-47, 5:13-22. As taught in *Kennedy*, the two application files (sus-a.exe and sus-b.exe) are tracked as part of the same file set and, in the combined method/system, the fact that sus-a.exe created sus-b.exe is recorded by *Kester's* remote workstation(s). Subsequently, sus-a.exe attempts to send sus-b.exe to another computer system and this event is also recorded. This related file behavior is transmitted as part of the application related data to the *Kester* server module and/or database factory.

212. A skilled artisan would have recognized implementation of *Kennedy's* file set tracking and behavior monitoring to be an obvious and predictable improvement to *Kester's* application file monitoring system/method. Indeed, a skilled artisan would have been motivated to apply the teachings of *Kennedy* to

enable *Kester's* application file monitoring system/method to include *Kennedy's* file set tracking and behavior monitoring to predictably improve the system/method's ability to detect suspicious behavior of different but related files. Furthermore, *Kennedy's* file set tracking and behavior monitoring would have made *Kester's* method/system more desirable and more effective in identifying malicious/rogue programs (such as malware), thereby improving similar malware detection method/systems in the same way.

213. A skilled artisan would have been readily familiar with the fact that "[a] successful worm spreads rapidly and can quickly damage many computer systems" (*McCorkendale*, [0005]; *Kennedy*, 1:21-23) and would have recognized that one of the ways that worms typically spread is by sending a file created or modified by the worm to another computer (*Kennedy*, 1:19-26, 5:15-22). Thus, a skilled artisan would have recognized that identifying when an application file attempts to send another file that it created or modified to another computer system as taught in *Kennedy* is a "hint" (in accordance with the teachings of *Kester*) that indicates that the application is infected by a worm and should be classified as malware. A skilled artisan, therefore, would have understood that implementation of file set tracking and behavior monitoring to determine whether related file behavior indicates that a piece of software is malware or not as taught by *Kennedy* is a predictable improvement to the *Kester* method/system.

214. In particular, *Kester* recognized a need to monitor and control application files, such as when employees "access server computers to download and execute rogue programs[.]" *Kester*, [0003], [0005]. Indeed, *Kester* already broadly explains that each remote workstation "monitors the ongoing behavior of the application" even after it determines that the application is allowed to run, which a skilled artisan would have understood encompasses monitoring the behavior of the set of files associated with the application. *Kester*, [0041]. *Kester* also describes that the application related data that is logged in the logging database and sent to the base station can "include additional data associated with the application" in addition to the specific examples disclosed. *Kester*, [0041]. A skilled artisan would have been motivated to apply the teachings of *Kennedy*, which teaches an effective method of tracking and monitoring different but related files and their behavior, to identify the types of rogue program threats *Kester* was designed to mitigate more effectively.

215. Because file set tracking and behavior monitoring was a well-known way to detect the presence of malware, there would have been a reasonable expectation of success configuring *Kester's* application file monitoring system/method to perform *Kennedy's* file set tracking and behavior monitoring. A skilled artisan would have understood that this combination of prior art elements would have been accomplished using known methods and with no change to their respective functions. For example, *Kester's* application file monitoring

system/method would still allow for monitoring application execution launch and network access behavior while also implementing *Kennedy's* known method of file set tracking and behavior monitoring to detect any suspicious behavior of related files, which would indicate the presence of a malicious/rogue program (malware).

216. Thus, in my opinion, *Kester* in view of *Honig* in further view of *Kennedy* renders Claim 3 obvious.

2. *Claim 4 – A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.*

217. *Kester* explains that "expected network activity can be determined from network activity by a different but related application" so, for example, "a later version of an application may share common access privileges with an earlier version of the same application." *Kester*, [0044], [0182]. *Kester* does not explain that an application's classification category is determined from the classification category (e.g., malware) of a "different but related application." However, modifying *Kester* to determine which category an application belongs to based on a different but related application would have been obvious to a skilled artisan. For example, *Kester* discloses that the "application and any related data" is displayed at the server module and/or database factory and that the related data is used to classify the application. *Kester*, [0134]-[0135], [0120]. A skilled artisan would have understood this disclosure of using "any related data" to classify an application

encompasses the classification category of a "different but related application." *Kester*, [0134]-[0135], [0120], [0044], [0182].

218. It would have been obvious to a skilled artisan to configure the *Kester's* malware classification process (*see* Claim 1, Element 1.3) based on the teachings of *Kester* and *Kennedy* such that if at least one other different but related application file (*other object*) in the same file set as the application file in question (*the object*) is classed as malware, the application file is classified as malware. The combined system/method classifies an application file as malware when at least one different but related application (i.e., another file in the same file set) is classed as malware.

219. For example, sus-a.exe directly or indirectly creates a new file, sus-b.exe, and sus-a.exe "sends more than a certain number of emails (e.g., N=10) within a certain time period is malicious (malware)." *Kennedy*, 4:19-47, 5:23-33. When *Kester* classifies sus-a.exe using the file set behavior monitoring of *Kennedy*, sus-a.exe is classified as malware because of its email transmission behavior. The combined method/system classifies sus-b.exe as malware because 1) sus-a.exe created sus-b.exe and 2) sus-a.exe has already been classified as malware—thereby classifying an application based on the related object's malware classification.

220. A skilled artisan would have been motivated to apply the teachings of *Kennedy* to enable *Kester's* application file monitoring system/method to include *Kennedy's* file set tracking and behavior monitoring to predictably improve the

system/method's ability to classify applications based on the classification of different but related files, as I described above. *See above*, ¶¶ 210-215 (further noting that this modification would have made *Kester's* method/system more desirable and more effective in identifying malicious/rogue programs (malware)). A skilled artisan would also have been motivated to implement the combined *Kester/Kennedy* system/method in this way to reduce the workload of the human reviewer, furthering one of the expressly stated goals of *Kester*—to "enhance productivity of the human reviewer." *Kester*, [0090].

221. A skilled artisan would have understood that this combination of prior art elements would have been accomplished using known methods. For example, a skilled artisan would have been well-familiar with the fact that "[a] successful worm spreads rapidly and can quickly damage many computer systems" (*McCorkendale*, [0005]; *Kennedy*, 1:21-23) and would have recognized that where an application has already been classified as malware (e.g., a worm or other malicious software), any file that the application touched (e.g., created and/or modified) is also likely infected (*Kennedy*, 1:11-25, 4:19-47, 5:12-33), as was well-known at the time. As explained in my opinions above, a skilled artisan would also have recognized that an application file (sus-b.exe) that was created by another application file (e.g., sus-a) that has already been classified as malware is a "hint" (in accordance with the teachings of *Kester*) that indicates that the application is infected (e.g., by a worm or

other malicious software) and should be classified as malware Therefore, there would have been a reasonable expectation of success configuring *Kester's* application file monitoring system/method to classify related applications as malware when one application in the related file set has been classed as malware.

222. Thus, in my opinion, *Kester* in view of *Honig* in further view of *Kennedy* renders Claim 4 obvious.

3. *Claim 5 – A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.*

223. *Kester* explains the importance of aggregating application related data from multiple remote computers and using the aggregated application related data to make application classification determinations. *See, e.g.*, *Kester*, [0045] ("In a preferred embodiment, the network activity from multiple workstations is uploaded to the application database factory 110."), [0119] (the application server module "merges and sorts the collected data including the frequency count with other workstation inventories" and merging and sorting "based on the application or any additional data associated with the application"), [0133] ("the collection data is merged and sorted" at the application database factory at step 1220), [0076] (using unexpected behavior to classify/reclassify the application). *See also* Claim 1, Elements 1.1, 1.2, 1.3.

224. It would have been obvious to a skilled artisan to configure *Kester's* system/method for monitoring application files such that each remote workstation additionally includes a file set tracking module as disclosed in *Kennedy* to log the set of files for each application in the logging database and each time a file in the set touches (e.g., creates, modifies, and/or reads) another file. *See* Claim 3. It also would have been obvious to implement the *Kester's* application server module and/or database factory to determine whether received data indicates that a program/application is malware based on the behavior and/or classification of different but related files. *See* Claims 3, 4. Based on the teachings of *Kester* to use aggregated application related data from multiple remote workstations to make application classification determinations, in the combined system/method, a skilled artisan would have understood the different but related files include the application file or similar application files on at least some of the remote workstations.

225. Thus, in my opinion, *Kester* in view of *Honig* in further view of *Kennedy* renders Claim 5 obvious.

4. *Claim 6 – A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.*

226. As I explained in my opinions above, *Kennedy's* file set tracking module 312 "tracks the behaviors of the files in the set and records each time a file [("parent")] in the set 'touches' [(e.g., creates)] another file [("child")] on the storage

device 108 or elsewhere in the computer system 100." *Kennedy*, 4:19-26. In addition to direct actions, "'touching' can be performed through indirect actions such as launching another process that then touches a file." *Id*., 4:31-33. Each touched file is added to the set of the file that touched it. *Id*., 4:33-34.

227. *Kennedy's* discussion of file set tracking functions, comprising tracking when one file touches/creates another file is consistent with the '250 Patent's description of "parent" and "child" objects. For example, the '250 Patent describes a process using "ancestral relationships" of an object to other objects (e.g., "which object created this object, which objects this object created") to make malware determinations. *See, e.g.*, '250 Patent, 11:3-23, 3:31-41, 17:15-67. The object that creates another object would be the "parent" or "father." The created object would be the "child" or "son." Thus, *Kennedy's* file sets include parent and/or child objects and/or otherwise process-related objects.

228. Thus, in my opinion, *Kester* in view of *Honig* in further view of *Kennedy* renders Claim 6 obvious.

5. *Claim 15 – Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the determining identifies relationships between the object and other objects.*

229. *See* Claim 3.

6. *Claim 16 – Apparatus according to claim 15, the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.*

230. *See* Claim 4.

7. *Claim 17 – Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least sonic [sic]of said remote computers.*

231. *See* Claim 5. I have been asked to assume that the claim term "sonic" is simply a typographical error that should properly read "some."

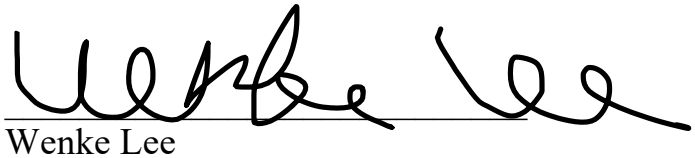
8. *Claim 18 – Apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.*

232. *See* Claim 6.

IV. CONCLUSION

233. 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: December 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

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) Georgia Institute of Technology	7/2015–7/2021
Director	Georgia Tech Information Security Center (GTISC) Georgia Institute of Technology	8/2012–6/2015
Professor	College of Computing Georgia Institute of Technology	2/2009–present
Associate Professor	College of Computing Georgia Institute of Technology	2/2005–2/2009
Assistant Professor	College of Computing Georgia Institute of Technology	8/2001–2/2005
Assistant Professor	Department of Computer Science North Carolina State University	8/1999–8/2001
Research Assistant	Department of Computer Science Columbia University	1994–1999
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

Semester/Year	Course	Number of Students	Comments
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

Semester/Year	Course	Number of Students	Comments
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

Semester/Year	Course	Number of Students	Comments
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 Satff 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 Noureddine, 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 Aller, 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. A11y 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 Nadji, 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 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 (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 Walfish. 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. Jonanthon 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 Honeypots. *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, Niranjan 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. <u>Other</u>

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 Analysis 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, Daejon, 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