UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SOPHOS LTD. and SOPHOS INC.,
Petitioners

v.

WEBROOT INC.,
Patent Owner.

Case No. IPR2023-00699
U.S. Patent No. 8,418,250

Declaration of Dr. Richard Newman
In Support of Petitioners in the
Petition for *Inter Partes* Review of U.S. Patent No. 8,418,250

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

Page

TABLE OF EXHIBITS IV

I. INTRODUCTION 1

II. BASIS FOR OPINIONS 2

A. Qualifications 2

B. Preparation for This Declaration 3

III. LEGAL STANDARDS 4

A. A Person of Ordinary Skill in the Art 4

B. Legal Standard for Prior Art 5

C. Claim Construction 6

D. Legal Standard for Obviousness 7

IV. TECHNOLOGY BACKGROUND 9

A. Overview of Malware 10

B. Overview of Malware-Detection Software 11

V. THE ’250 PATENT 15

A. Overview of the ’250 Patent 15

B. Overview of the Challenged Claims 18

VI. PRIOR ART CONSIDERED 22

A. Identification of the Primary Prior-art references 22

VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART 23

A. Claim Construction 23

B. Application of the Prior Art to the Claims 23

C. Ground 1: Halperin in View of Mankins Renders Obvious Claims 1, 12-14, and 24 24

1. Prior Art Overview 24

2. Motivation to Combine and Reasonable Expectation of Success 27
3. Analysis 33
a. Independent Claim 1 33
b. Independent Claim 13 54
c. Dependent Claims 2 and 14 54
d. Dependent Claims 12 and 24 55
D. Ground 2: Halperin in View of Kirby and Mankins Renders Obvious Claims 1-30 57
1. Prior Art Overview 57
2. Motivation to Combine and Reasonable Expectation of Success 58
3. Analysis 60
a. Independent Claim 1 60
b. Independent Claim 13 62
c. Dependent Claims 2 and 14 63
d. Dependent Claims 3 and 15 63
e. Dependent Claims 4 and 16 64
f. Dependent Claims 5 and 17 65
g. Dependent Claims 6 and 18 67
h. Dependent Claims 7 and 19 69
i. Dependent Claims 8 and 20 70
j. Dependent Claims 9 and 21 71
k. Dependent Claims 10 and 22 72
l. Dependent Claims 11 and 23 73
m. Dependent Claims 12 and 24 75
n. Independent Claim 25 75
o. Dependent Claim 26 77
p. Dependent Claim 27 77

q. Dependent Claim 28 77
r. Dependent Claim 29 77
s. Dependent Claim 30 77
VIII. SECONDARY CONSIDERATIONS 78
IX. RIGHT TO SUPPLEMENT 78
X. CONCLUSION 79

TABLE OF EXHIBITS

Exhibit No.	Document
1001	U.S. Patent No. 8,418,250 ("the '250 Patent")
1002	File History for U.S. Patent No. 8,418,250
1003	Complaint, *Webroot Inc, et al. v. Sophos Ltd.*, No. 6:22-cv-240 (W.D. Tex.)
1004	Declaration of Dr. Richard Newman
1005	Curriculum Vitae of Dr. Richard Newman
1006	U.S. Patent Pub. No. 2004/0111632 to Halperin ("Halperin")
1007	U.S. Patent No. 7,694,150 to Kirby ("Kirby")
1008	U.S. Patent No. 7,900,194 to Mankins ("Mankins")
1009	Stipulation Waiving IPR Claims in District Court Litigation
1010	S. Forrest *et al.*, *Computer Immunology*, Comms. of the ACM, Vol. 40, No. 10, 88–96 (1997)
1011	A. Snoeren *et al.*, *Single-Packet IP Traceback*, IEEE/ACM Trans. on Networking (TON), Vol. 10, No. 6, 721–34 (2002)
1012	S. Hofmeyr, *Intrusion Detection Using Sequences of System Calls*, J. Computer Security, Vol. 6, Issue 3, 151–180 (Aug. 1998)
1013	S. Forrest *et al.*, *A Sense of Self for Unix Processes*, Proc. 1996 IEEE Symp. on Security and Privacy, 120–28 (1996)
1014	Order Granting Fourth Amended Scheduling Order, Dkt. 160, *Webroot, Inc., et al. v. AO Kaspersky Lab*, No. 6:22-cv-00243-ADA-DTG (W.D. Tex.) (Jan. 22, 2023)
1015	Excerpt from Transcript of Motions Hearing in *Webroot, Inc., et al. v. AO Kaspersky Lab*, No. 6:22-cv-00243-ADA-DTG (W.D. Tex.) (Dec. 9, 2022)
1016	*Threat Graphs examples: Malware detection*, Sophos Ltd., KB-000036359, https://support.sophos.com/support/s/article/KB-000036359?language=en_US (Feb. 23, 2023)

I. INTRODUCTION

1. I, Dr. Richard Newman, have been retained by Gibson, Dunn & Crutcher LLP on behalf of Sophos Ltd. and Sophos Inc. ("Petitioners") as an independent expert in this Petition for *inter partes* review by the U.S. Patent and Trademark Office. As part of my engagement, I have been asked to provide analysis and expert opinions on the following topics: (a) the disclosure of the U.S. Patent No. 8,418,250 ("the '250 Patent"); and (b) the validity of claims 1-30.

2. It is my opinion that claims 1-30 of '250 Patent ("the Challenged Claims") are invalid in view of the prior art I discuss in this Declaration. My opinion is that the prior art discloses the invention claimed in the Challenged Claims, so the Challenged Claims are invalid as obvious over the prior art.

3. I am being paid for my work on this case at my standard consulting rate and being reimbursed for my expenses. My compensation is not contingent upon the results of my study, the substance of my testimony, or the outcome of this case.

4. Based upon the facts and information available to me to date, I detail my opinions relevant to this petition for IPR. I am prepared to provide expert testimony on the opinions I express if asked about those issues by the Board or others.

5. In this Declaration I discuss my own work, teachings, and knowledge of the state of the art in the relevant time period, and I cite technical literature (like handbooks and textbooks) to demonstrate the state of the art in the relevant period.

6. Throughout this declaration, I refer to specific line numbers, paragraph numbers, or page numbers of the relevant prior art or other technical documents. Those citations might not be exhaustive—there may be more evidence to support a particular proposition than I cite here in this Declaration.

II. BASIS FOR OPINIONS

A. Qualifications

7. I recently retired from the faculty of the University of Florida's Computer and Information Science and Engineering Department. I completed my Ph.D. in Computer Science from University of Rochester Computer Science Department in Rochester, NY, in 1986, after I earned my M.S. there in 1983. I earned my BA in Mathematics from New College in Sarasota, FL, and attended Eckerd College in St. Petersburg, FL.

8. I introduced the graduate-level course on Computer and Network Security at the University of Florida. I have also taught graduate-level courses on networks, operating systems, distributed systems, and computer and information security; and special-topics courses in cryptography, cryptographic protocols, anonymity, the theory of security, and cybersecurity. I also chaired the university-

wide Networking and Telecommunications Standing Committee in the mid-1990s, which considered practical issues in network provision, management, and security for the UF campus-wide network and telecommunications facilities.

9. My publications include topics in security models, access control, steganography, covert channels, traffic analysis, and cryptographic protocols. I am co-inventor of 17 patents, most of which disclose shared-networking functions. I am a member of the Institute of Electrical and Electronics Engineers (IEEE) and ACM (Association for Computing Machinery).

10. I have technical expertise in computer and network security; cryptography and cryptographic protocols; distributed systems; network protocols, topologies, and architecture; and powerline communications. My full qualifications and experience are set forth in my Curriculum Vitae, which is attached as Exhibit 1004.

B. Preparation for This Declaration

11. My findings contained in this declaration are based on my education, research, experience, and background in the fields of network computing and cybersecurity, as well as my review of prior-art references and other documents disclosed in this declaration. In forming my opinions, I have studied and considered the ’250 Patent, the file history of the ’250 Patent, the prior-art references cited in this declaration, and the other references I cite or refer to in this report.

12. I understand that this declaration will be submitted in support of a Petition requesting *inter partes* review of the ’250 Patent.

III. Legal Standards

13. Counsel for the Petitioners have told me about the following legal standards for this case, which I have adhered to and applied throughout my analysis.

A. A Person of Ordinary Skill in the Art

14. Counsel for the Petitioners have told me that a person of ordinary skill in the art, or a “POSITA,” is a hypothetical person from whose viewpoint the prior art is analyzed (without the benefit of hindsight). A POSITA uses conventional wisdom in the art but does not try to innovate, either by extraordinary insights or by systematic research.

15. Counsel for the Petitioners have also told me that the hypothetical POSITA has knowledge of all prior-art references that are sufficiently related to one another and to the pertinent art, and to know of all prior art in the fields reasonably pertinent to the particular problem that the claimed invention purports to address.

16. In evaluating the prior-art references and other material, I have used the perspective of a POSITA in the field to which the patent is related as of the time of the patent’s priority date. I understand that a POSITA is presumed to be aware of all pertinent prior art and the conventional wisdom in the art and is a person of

ordinary creativity. I have applied this standard in my analysis throughout my declaration.

17. The ’250 Patent is entitled “Methods and Apparatus for Dealing With Malware.” A POSITA in the June 2005 timeframe, which I am informed by Counsel is the earliest possible priority date of the ’250 Patent, would have had at least a Bachelor’s degree in Computer Science, Computer Engineering, Electrical Engineering, or an equivalent discipline, and at least two years’ worth of experience developing dynamic malware detection and computer networking or network architecture, or a Master’s degree in a related field. Or, instead of the education requirements, a POSITA could have four or more years of work experience in computer networking and in the development of client-server systems and dynamic malware detection.

18. I meet the criteria described above and I am a person with at least ordinary skill in the art pertaining to the ’250 Patent. I would have qualified as such a person by at least June 2005, which I understand is the earliest priority date of the ’250 Patent.

B. <u>Legal Standard for Prior Art</u>

19. Counsel for the Petitioners have told me that a patent, published patent application, or other publication, must first qualify as prior art before it can be used to invalidate a patent claim.

20. Counsel have told me that a U.S. or foreign patent qualifies as prior art to a patent claim if the date of application, issuance, or publication of the prior-art patent is before the purported date of invention of the patent claim.

21. Counsel have also told me that a printed publication (like a technical publication, a magazine article, or newsgroup post) qualifies as prior art to a patent claim if the date of publication is before the purported date of invention of the patent claim. Counsel have also told me that a U.S. patent qualifies as prior art to a patent claim if the application for the prior-art patent was filed in the United States before the purported date of invention of the patent claim. Finally, Counsel have told me that, if a U.S. Patent incorporates another patent or document by reference, it is considered to be part of that same reference.

C. Claim Construction

22. Counsel have told me that in an IPR the Board uses the same standard for construing the challenged claims as during claim construction in district court civil actions. Counsel have also told me that, in the district court context and in this IPR, the words of a claim are generally given their ordinary and customary meaning—that is, the meaning a term would have to a person of ordinary skill in the relevant art at the time of the invention. The time of invention is generally the time the application was filed, although in this case Counsel have told me that Patent Owner has asserted an invention date of June 2005. Counsel have asked me to use

June 2005 for my analysis in this Declaration, so I did. But I do not necessarily agree that June 2005 is the correct priority date, and I do not mean to convey by using that date that I think it is the correct date.

23. Counsel have also told me that the specification informs the proper interpretation of a claim. This is because a POSITA is assumed to have read the claim term not only in the context of the particular claim in which the disputed term appears, but in the context of the entire patent, including the specification. Counsel have also told me that the prosecution history and extrinsic evidence concerning relevant scientific principles, the meaning of technical terms, and the state of the art are also considered in ascertaining the meaning of claim terms.

D. Legal Standard for Obviousness

24. Counsel have told me that an invention may be obvious, even if it is not identically disclosed or described in a single prior-art reference, as long as the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time the invention was made.

25. Counsel have also told me that the Supreme Court has told the PTO to be cautious granting a patent based on the combination of elements found in prior art, because a patent for a design that only combines old elements but does not actually change their respective functions removes an already-known design from

the public and gives it exclusively to the patentee. The combination of familiar elements according to known methods is likely to be obvious when it does no more than yield predictable results.

26. Counsel have also told me that when a work is available in one field of endeavor, design incentives and other market forces can prompt variations of it, either in the same field or a different one. If a POSITA can implement a predictable variation, the invention is probably obvious. For the same reason, if a technique has been used to improve one device, and a POSITA would recognize that the technique would improve similar devices in the same way, using the technique is obvious unless its actual application is beyond the POSITA's skill. It is my understanding, in other words, that a court has to determine whether the improvement is more than the predictable use of prior-art elements according to their established functions. Counsel have told me that that this standard is typical and it is not confined to any particular formalistic conception.

27. Counsel have also told me that an obviousness analysis should consider the scope and content of the prior art, differences between the prior art and the claims of the challenged patent claims, and the level of ordinary skill in the art. Secondary considerations of obviousness or non-obviousness, where supplied, must also be considered. Secondary considerations can include simultaneous or near-simultaneous invention, commercial success enjoyed by devices practicing the

patented invention, industry praise for the patented invention, copying by others, and the existence of a long-felt but unsatisfied need for the invention. I am also informed by Counsel that there must be a nexus between these factors and the claims themselves for these secondary considerations to have an impact on the analysis of obviousness.

28. Counsel have told me that there generally must be a motivation to combine the references in order to find obviousness. This motivation or reason to combine can arise from the references themselves, changing market dynamics, or from simple common sense.

29. I am further informed by Counsel that a single reference can render a patent claim obvious, even if that reference does not explicitly or inherently disclose every element of the claim.

30. To determine whether there was an apparent reason to combine the known elements in the way a patent claims, it will often be necessary to look to interrelated teachings of multiple patents; to the effects of demands known to the design community or present in the marketplace; and to a POSITA's background knowledge.

IV. TECHNOLOGY BACKGROUND

31. To put in context my opinions on the validity of the claims of the '250 Patent, in this section I provide some background on malware and malware-detection

software. All of the concepts discussed in this section were well-known and widely used well prior to the June 30, 2005, the priority date I understand that Patent Owner asserts for the ’250 Patent.

A. <u>Overview of Malware</u>

32. Malware, or malicious software, is software that targets information on a computer or network in order to misuse (*e.g.*, by modifying, stealing, or destroying) the data. Malware may come in the form of a “virus” that inserts its own code into existing programs to destroy, encrypt, modify, or exfiltrate data, and or programs; slow system resources; or log keystrokes, screenshots, and or other user actions; and to reproduce by infecting other files and spread to other computers to repeat the process. Malware may also exist as a “worm” that can spread without attaching itself to existing files, and can (for instance) delete, encrypt, or steal data. It may also have broader goals unrelated to specific data; for instance, it may exhaust resources.

33. Worms generally do not require input to reproduce, and they typically spread to other computers on a network. They often do this by exploiting a vulnerability on the target machine over a network, then transferring a file (executable or script) to the target machine, and then executing the transferred file. This creates a new child process of the service or process that has been compromised. In some cases, this child process transfers or creates another file with a more

complex program or script, which it then executes, creating a child of the child process. One danger of runaway worm programs is that they can exhaust the ability of the host operating system to create new processes due to repeated infections creating too many child processes. "Trojan horses" are malware software programs that appear harmless or legitimate so that an end user will unwittingly grant them access to a computer or system. For example, a Trojan horse software may appear as a regular operating-system update from Microsoft, but once installed, it may damage, disrupt, steal, infect, etc., the data or programs on a computer or network.

34. Malware enters a computer system from an external source, usually the internet or some external storage device (e.g., external hard drive or flash drive). Malware may be hidden inside software bundles or a storage device's firmware, or it may be disguised as a recognizable program.

B. <u>Overview of Malware-Detection Software</u>

35. Malware detection usually involves analyzing a signature of a suspect object. A signature, represented by a sequence of bytes, is a characteristic of the malware (or other object), which identifies that object with good specificity and that uses much less space than the full object itself. The signature may be based on extrinsic metadata (e.g., size, name, date of creation or modification, imported or exported functions, printable strings, etc.) or intrinsic (certain portions of the file or code, or all of it, or a pattern somewhere within the object), or a digest derived

mathematically by applying a function to some or all of the object and or its metadata. The purpose of a signature is to identify a particular object (e.g., malware, or alternatively, legitimate software from a reputable source) with high reliability. Malware-detection-software organizations compile the signatures for known malware and store them in a database. That way, the software can generate signatures for objects introduced into computer systems and compare the generated signatures to those in the database. If the signature is found in the database, the object is classified as malware and isolated or otherwise handled.

36. This traditional method is effective against known malware, but it has drawbacks: It relies on an up-to-date, exhaustive database. Brand-new malware can compromise a computer or network before the software provider is able to update the database with the signature for the new malware. And if the signature is not in the database, the new malware will not be detected and may "infect" the computer or network. So the reliability of this traditional method depends in large part on up-to-date databases.

37. Some malware creators may also hide the attributes of their malicious code that malware-detection software could use to create its signature, allowing malware to evade detection and infect the computer or network (e.g., polymorphic malware, which changes its appearance but not its functionality when it spreads its infection). So even if a particular malware's signature is already known and stored

in the database, a variant of it that "looks" different but functions in essentially the same way may be able to evade detection and infect its target.

38. Malware-detection software may also use behavior analysis. Behavior analysis assesses a suspect object's actual or attempted behavior (i.e., what it has done or tries to do within the computer or system if it is allowed entry). Behavior-analyzing software might allow a process to run on a user's system, or it may run the object in a particular environment and in either case assess whether the object's activity is malicious or benign. The particular environment may be a "sand-box" in which the process runs with restricted access privileges, perhaps on the real system, or the environment may be a simulated or emulated environment in which the suspect object can do no harm to the real system. The advantage of behavior-analysis detection is that it works for objects that have not been previously detected because it analyzes the actions an object takes or tries to take, rather than statically analyzing the object's characteristics (e.g., code or metadata).

39. Behavior-based analysis takes two general forms. One form specifies suspicious or known bad behaviors often taken by malware (e.g., changing the registry contents, modifying executable files, sending numerous emails without user input, sending emails with certain types of attachments, etc.). Often a single isolated action is insufficient to indicate malware reliably, since some of the suspicious actions may be performed by legitimate programs as well, so the system may be set

to diagnose an object as malware only when it performs multiple suspicious actions. Some types of malware may be positively identified by specific sequences of actions (e.g., deleting or encrypting files) while previously unknown malware might only be identified as malicious but not associated with a known malware instance.

40. The second form of behavior analysis attempts to define "normal" or "acceptable" behavior, and requires that an uninfected system be monitored and its behaviors analyzed to arrive at a sufficiently complete collection of behaviors that benign processes exhibit. When an unknown program (or a modified version of a known program) executes, either it or the system as a whole may be monitored for behavior to determine whether or not the attempted or actual behavior is "normal." Known processes, particularly those that may be vulnerable to intrusion attacks, may also be monitored for behavior that is specific to them, more tightly defining "normal" for the particular program or service. This approach may rely upon statistical analysis, specific sequences of actions, or other means to detect anomalous behavior and to flag it. Anomaly detection thus may include acceptability parameters. If anomaly-detection software with acceptability parameters discovers that an object has performed or has attempted to perform an activity or activities outside those parameters, it may classify that object as malware. In order to avoid excessive false positives (i.e., identifying a legitimate process as malware), it is often necessary to allow for some anomalous behavior to occur without raising an alert,

and so some means of setting acceptable levels of anomalous behavior must be employed in such systems.

41. Malware-detection systems may be distributed across a network. Distributed malware detection confers certain advantages, including greater consistency (because a central or authoritative version of an object may be made available to all connected devices), reliability, speed of detection, scalability, and ease of maintenance.

V. The ’250 Patent

A. Overview of the ’250 Patent

42. The ’250 Patent claims to ameliorate some of the deficiencies in malware detection discussed above with a process of identifying, at a base computer, objects at remote computers that may be malicious. The ’250 Patent notes that anti-malware programs often use databases that store the unique signatures of known malicious software, and the software developers regularly update these databases with new entries as new malware is discovered. Ex. 1001 at 1:18-29.

43. The ’250 Patent also notes that prior-art anti-malware software may use heuristics to identify malware; for instance, it may gather data on multiple instances of a particular object across different remote computers and correlate those different data points (regarding, e.g., name, location, vendor, product, and version) to create a score. *Id.* at 10:41-47. Other processes, as the ’250 Patent notes, may

identify and classify as malicious or benign any objects related to the suspect object (e.g., objects that the suspect object created, or the object that created the suspect object, etc.). *Id.* at 11:3-58.

44. According to the ’250 Patent, users are at risk when there is a delay in updating the service providers’ databases, *id.* at 1:30-37, but any deficiencies in the processes used by prior-art malware-detection systems can be avoided with a system of remote computers set up to communicate with a base computer. *Id.* at 3:8-12. The network uses data from the object and others like it across the remote computers to determine whether the object is malware.

45. Figure 1 (reproduced below) describes the system:

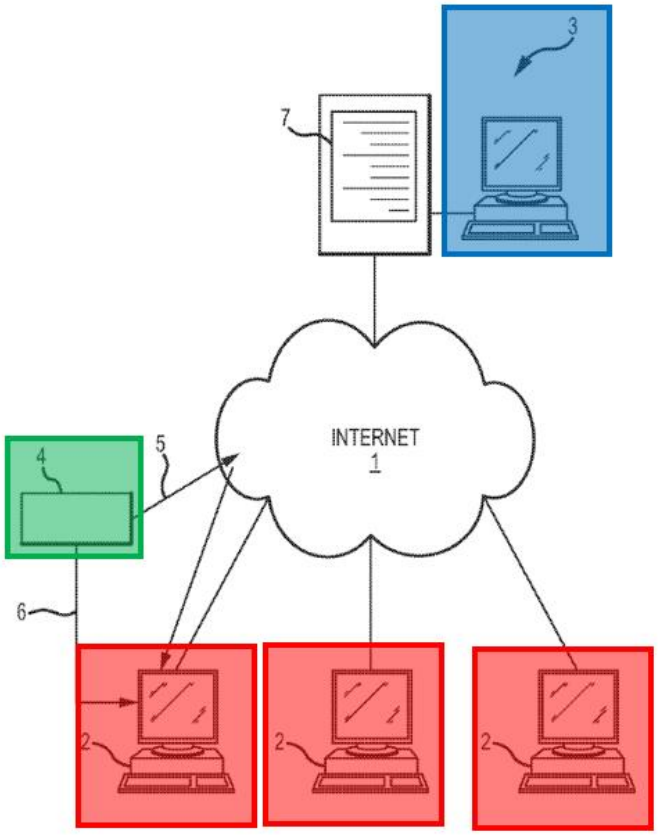

FIG.1

46. In the ’250 Patent, **remote computers 2** communicate with a **base or central computer 3**. *Id.* at 7:54-63. When the system discovers a new **computer object 4** on a **remote computer 2**, that **remote computer 2** sends data representing **computer object 4** to the **base computer 3**. *Id.* at 8:44-50. **Base computer 3** receives and correlates data from multiple **remote computers 2**. *Id.* at 10:41-44. The system can then compare the correlated data to data that corresponds to another object or entity. “[T]he comparing identifies relationships between the object and

other objects" and the unknown computer object classified as malware on the basis of that comparison. *Id.* at 3:31-35, 10:49-11:2.

B. Overview of the Challenged Claims

47. There are three independent claims of the '250 Patent: claims 1, 13, and 25. There are 27 dependent claims. I am informed by Counsel that Petitioners are challenging all 30 claims. I have analyzed each of these claims reproduced below.

48. For the reasons I detail below, it is my opinion that each of these claims is invalid as obvious over the prior art discussed below.

Independent Claim Limitations
[1P] A method of classifying a computer object as malware, the method comprising:
[1A] 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;
[1B] determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;
[1C] classifying the computer object as malware when the data indicates that the computer object is malware;
[1D] when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware;
[1E] 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;

[1F] running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers;
[1G] allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;
[1H] disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and,
[1I] reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.
[13] Claim 13 is the apparatus version of Claim 1.
[25P] 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:
[25A] providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;
[25B] 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;
[25C] 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,
[25D] 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;
[25E] executing the computer object at the remote computer; automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;
[25F] allowing the object to continue to run when behaviour of the computer object is permitted by the mask;

[25G] disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and,
[25H] reclassifying the object as malware when the actual monitored behaviour extends beyond [t]hat permitted by the mask,
[25I] wherein the base computer is arranged to receive information about the behaviour of the object running on one or more remote computers**,**
[25J] 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.
Claim Limitations Dependent From Claims 1 and 13
[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. **[14]** Claim 14 is the apparatus version of Claim 2.
[3] A method according to claim 1, wherein determining identifies relationships between the object and other objects. **[15]** Claim 15 is the apparatus version of Claim 3.
[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. **[16]** Claim 16 is the apparatus version of Claim 4.
[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. **[17]** Claim 17 is the apparatus version of Claim 5.
[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. **[18]** Claim 18 is the apparatus version of Claim 6.

[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. **[19]** Claim 19 is the apparatus version of Claim 7.
[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. **[20]** Claim 20 is the apparatus version of Claim 8.
[9] A method according to claim 7, wherein the key has at least one component that represents data about said object. **[21]** Claim 21 is the apparatus version of Claim 9.
[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. **[22]** Claim 22 is the apparatus version of Claim 10.
[11] A method according to claim 7, wherein the key has at least one component that represents the physical size of the object. **[23]** Claim 23 is the apparatus version of Claim 11.
[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. **[24]** Claim 24 is the apparatus version of Claim 12.
Claim Limitations Dependent From Claim 25
[26] Claim 26 includes the same additional limitations as Claims 8 and 20.
[27] Claim 27 includes the same additional limitations as Claims 9 and 21.
[28] Claim 28 includes the same additional limitations as Claims 10 and 22.

[29] Claim 29 includes the same additional limitations as Claims 11 and 23.
[30] A non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

VI. PRIOR ART CONSIDERED

49. I provide below prior-art references with exemplary citations identifying the relevant features related to the claim language of the Challenged Claims. I have also described combinations of these prior-art references that, when combined, render the Challenged Claims of the ’250 Patent obvious. In my opinion the Challenged Claims of the ’250 Patent are rendered obvious in light of the prior art specifically discussed below.

50. The citations presented in the body of this declaration correspond to the teachings disclosed in exemplary items of prior art that identify: (1) the problem(s) confronted by one of ordinary skill in the art; (2) an express suggestion to make one or more of the combinations; (3) an implicit suggestion to make one or more of the combinations; or (4) the knowledge of those skilled in the art as to the applicable field of technology at the relevant timeframe.

A. Identification of the Primary Prior-art references

51. My full element-by-element analysis is provided in more detail below (VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART). It is my opinion that a person of ordinary skill would have considered at least

the prior-art references in the following table. As mentioned before, it is my opinion that the references summarized below disclose or suggest the features of the Challenged Claims of the ’250 Patent.

Prior-Art Reference	Filing / Priority Date	Exhibit No.
U.S. Patent Pub. No. 2004/0111632 to Halperin (“Halperin”)	May 5, 2003	1006
U.S. Patent No. 7,694,150 to Kirby (“Kirby”)	June 22, 2004	1007
U.S. Patent No. 7,900,194 to Mankins (“Mankins”)	Mar. 25, 2004	1008

VII. ANALYSIS OF THE CHALLENGED CLAIMS AGAINST THE PRIOR ART

A. Claim Construction

52. I apply the plain and ordinary meaning for each term below, as they would have been understood by a POSITA at the priority date of the patent.

53. Where I address the preamble of a claim, I am not opining that the preamble is limiting or not as a matter of claim construction. Instead I am opining that, to the extent the preamble is limiting, it is disclosed and/or rendered obvious for the reasons I describe below.

B. Application of the Prior Art to the Claims

54. Based upon my review of the prior art, it is my opinion that the subject matter of each of the Challenged Claims is anticipated by, or would have been obvious to one of ordinary skill in the art in view of the known prior-art references.

55. I am informed by Counsel that each of the prior-art references analyzed in this declaration meets the legal requirements for prior art to the ’250 Patent. I have performed my analysis with this understanding.

56. Based upon my analysis, it is my opinion that the Challenged Claims of the ’250 Patent are anticipated or rendered obvious by the following prior art:

Ground	Claims	Opinions
1	1, 2, 12-14, 24	Obvious under 35 U.S.C. § 103 over Halperin in view of Mankins
2	1-30	Obvious under 35 U.S.C. § 103 over Halperin in view of Mankins and Kirby

C. Ground 1: Halperin in View of Mankins Renders Obvious Claims 1, 12-14, and 24

57. Halperin correlates data received from multiple remote computers about a computer object in order to classify the object as malware or non-malware:

> After collecting information regarding target behavior detected at two or more of computers 500, server 502 may then correlate the presence of target behavior detected at two or more of computers 500 in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers.

Ex. 1006, [0047].

1. Prior Art Overview

a. Halperin

58. Halperin, like the ’250 Patent, discloses “detecting malicious activity in a computer network,” Ex. 1006, Abstract, “by correlating behavior and/or data

from different computers." Ex. 1006, [0037]. Halperin also "overcomes disadvantages of the prior art." *Id.*, [0007]. According to Halperin, previous designs were "virus scanners" that "operate by searching a computer for tell-tale byte sequences known as 'signatures' that exist in known viruses," Ex. 1006, [0005], and "antivirus software that employs heuristic techniques to identify typical virus behavior . . . and then identifying any deviation from such behavior." Ex. 1006, [0006].

59. Figure 5 of Halperin (shown below) graphically depicts the system in Halperin:

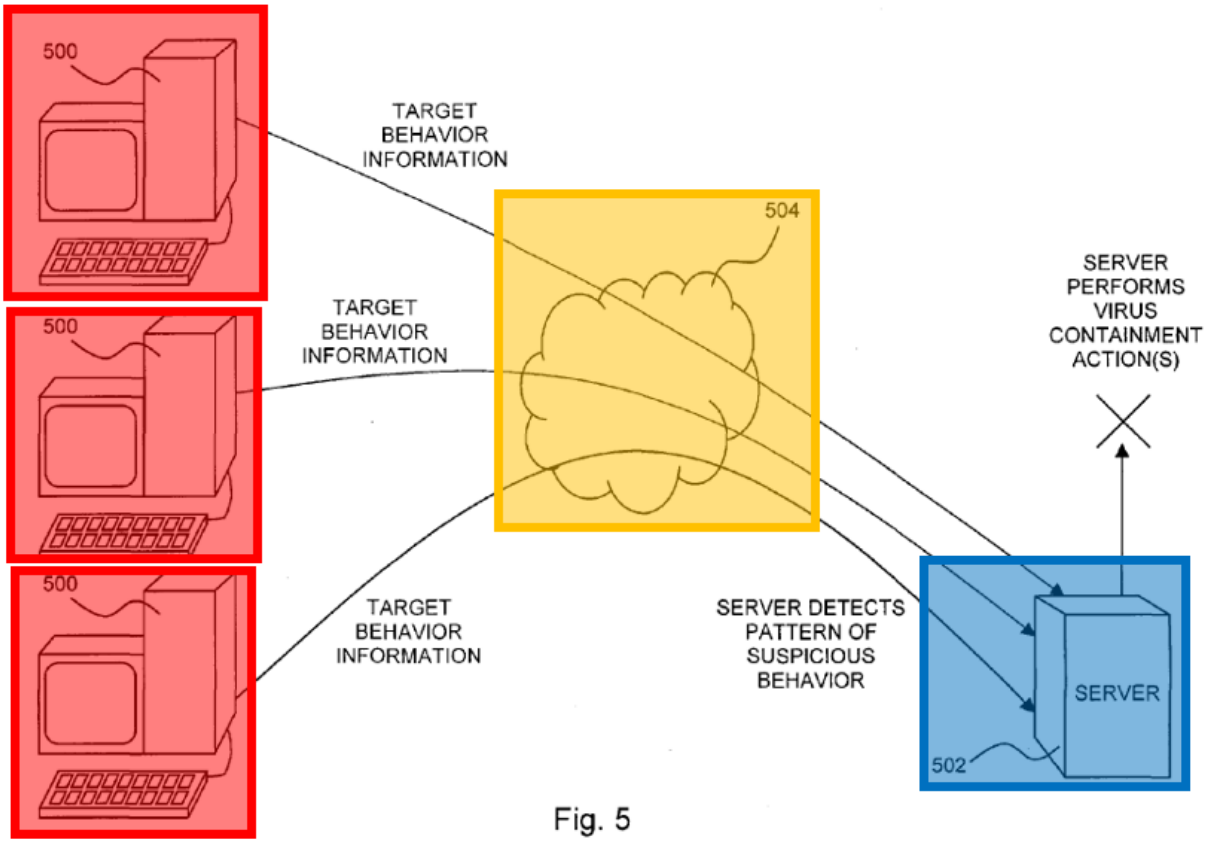

Fig. 5

60. "In the system of Fig. 5 one or more **computers 500** are shown, being configured to communicate with a **server 502** via a **network 504**, such as the Internet." Ex. 1006, [0036]. **Server 502** "may be configured with such target behavior profiles and may detect associated target behavior at **computers 500** using conventional techniques." *Id.*, [0047]. "After collecting information regarding target behavior detected at two or more of **computers 500**, **server 502** may then correlate the presence of target behavior detected at two or more of **computers 500** in order to determine whether the correlated target behavior corresponds to a predefined suspicious behavior pattern." *Id.* Halperin then uses a "predefined suspicious behavior pattern of target behavior," *id.*, [0047], to compare the correlated data, and by comparing the correlated data with the "predefined suspicious behavior pattern of target behavior," Halperin determines "that a computer virus may have infected those computers." *Id.*

b. Mankins

61. Mankins discloses a "system and method for kernel-based intrusion detection using Bloom filters." Ex. 1008 at 3:18-19. In Mankins, "normal (i.e., expected) system-call behavior is defined by monitoring system-call behavior during operation of the computer program under training conditions (e.g., when the computer program is not subject to intrusion attacks). *Id.* at 3:37-40. "The intrusion detection system then uses a Bloom filter to generate a Bloom filter data object

containing data representative of the normal system-call behavior." *Id.* at 3:40-43. "The predefined Bloom filter data is representative of 'normal' computer program behavior, which refers to behavior that is expected during operation of the computer programs when not subject to intrusion attempts." *Id.* at 3:26-30. "Anomalous behavior is identified when the actual system-call behavior does not have a match in the Bloom filter data object." *Id.* at 3:53-55.

2. Motivation to Combine and Reasonable Expectation of Success

62. A POSITA would have found it obvious to incorporate Mankins's Bloom filter data object into Halperin's malicious activity detection system. Doing so would allow the system to detect malware using expected system-call behavior alongside, or instead of Halperin's suspicious event sensors. Halperin and Mankins are both related to techniques for malware detection, and both use behavioral approaches that observe a computer object's or program's actions and compare those actions to known suspicious events, as in Halperin, or to known normal, acceptable behavior, as in Mankins. Within the behavioral approach, combining detection of known "suspicious" behavior with detection of "abnormal" behavior outside the "mask" of acceptable behavior increases the chances that malware is detected and detected early.

63. Malware-intrusion detection systems used sequences of system calls by the mid-1990s. *See, e.g.*, Ex. 1012 (describing a method for anomaly intrusion detection at the process level using "short sequences of system calls"); Ex. 1013 (Forrest et al.) at 1 ("A method for anomaly detection is introduced in which 'normal' is defined by short-range correlations in a process' system calls."). A POSITA would have been able to predictably use a well-known prior-art element in a way that made use of its established function.

64. It would have been obvious to a POSITA to implement Mankins's system in a distributed configuration (for instance, with the architecture of Halperin, which uses remote computers and a server), notwithstanding Mankins's exemplary-configuration implementation on a single computer 110 at the operating-system kernel level. Mankins even says as much: its "intrusion detection system 100 may be implemented *in a distributed fashion over a network of computing devices*." Ex. 1008 at 4:29-30 (emphasis added); *see also id.* at 4:50-52 ("In many embodiments, the computer 110 is in the form of one or more network servers." (emphasis added)). And Mankins also teaches "network embodiments of the intrusion detection system" in which "multiple network devices are configured to work together to identify intrusion attacks." Mankins even explains one aspect of the utility of a networked configuration:

> [W]hen several network devices generate separate Bloom filter data objects 180, are monitored together, and are subject only to attacks that affect each of the network devices within a small amount of time, it is likely that at least one of the network devices would detect a false positive that is produced by another device. Thus, the sizes of the Bloom filter data objects 180 may be reduced because the different network devices can work together to identify false positives.

Id. at 10:53-64.

65. As Mankins explains, a networked configuration would have been a "[c]onventional system-call-based intrusion detection technique[]" where "tracked system-call sequences are inserted into a *database* to form a profile of system-call sequences that are considered to be normal operations of the particular program," and where "the monitored system-call sequences" that are "tracked in the OS kernel" are compared "with data of a *profile database stored outside of the* OS." *Id.*, 1:60-2:21 (emphases added).

66. A POSITA would have found it obvious to modify Halperin's system in a way that implemented Mankins's Bloom filter system: A POSITA would have known that, just as the Halperin system divides its behavioral approach between the remote computers 500 and the central server 502, the Mankins system's functions could be divided between the computers 500 and the server 502. In one approach,

each remote computer could generate hashes of an object's observed behavior. Then, if the object demonstrated no suspicious or otherwise unacceptable behavior during Halperin's comparison of correlated target behavior data its predefined suspicious behavior pattern, the computers 500 would be able to send the monitored behavior's hashes (which would include the normal system-call patterns) to the server 502, which could create a Bloom filter data object, or update an existing one, for the object/program.

67. In a second approach, the OS kernel at each of the computers 500 could collect sequences of system calls made by the object/program in a buffer and maintain the Bloom filter object for the program at each computer 500, as described in the preferred embodiment of Mankins: "By embedding the Bloom filter data object in the OS kernel, the intrusion detection system is able to compare actual program behavior monitored in the OS kernel with predefined normal program behavior, without having to access data outside of the OS kernel." Ex. 1008 at 3:63-4:1. Subsequently, the Bloom filter objects could be sent to and collected at the server 502, and an updated Bloom filter object based on the data collected from each remote computer 500 could be distributed back to each computer 500, as in the networked embodiment of Mankins. *See id.* at 10:53-64 ("[W]hen several network devices generate separate Bloom filter data objects 180, [and] are monitored

together . . . the different network devices can work together to identify false positives.”).

68. The computers 500 could then monitor the program’s behavior by, for instance, observing system-call sequences and comparing them to the up-to-date Bloom filter data object. It could do so right on the remote computer, or it could send the hashed system-call-sequence values to the server 502, where they could be compared to the Bloom filter data object stored at external server 502. Alternatively, the kernel could continue to cache the sequence of system calls made by the process in a relatively small buffer and send the buffer contents either periodically or when the buffer became sufficiently full. In either approach, the Mankins Bloom filter process would begin where the Halperin “predefined suspicious behavior pattern” check did not identify malware. In both approaches, the combined system would follow Mankins’s teachings to use the monitored behavior containing the “normal” system call patterns to generate a (or update an existing) Bloom filter data object for that program, and monitor the behavior of the program (e.g., by monitoring system call sequences) and compare the behavior to the Bloom filter data object directly at the remote computer, or transmit the information for comparison against a Bloom filter data object stored at the server 502.

69. The additional cost of performing the extra test would be relatively small, since Halperin already has the monitoring system in place, and so this would

be an attractive trade-off for a POSITA. Indeed, Mankins specifically notes that its approach has low memory requirements: "in many computers, the amount of memory required by a particular Bloom filter data object 180 (e.g., 3 KB) will be a small memory outlay that may easily fit in the process structure of the OS kernel 120, thereby allowing real-time detection of intrusion attacks." Ex. 1008 at 10:48-52.

70. Moreover, Halperin recognizes that anomaly detection (such as the technique described in Mankins) was a well-established approach to malware detection. "Another approach involves using antivirus software that employs heuristic techniques to identify typical virus behavior by *characterizing legitimate software behavior* and then identifying any deviation from such behavior." Ex. 1006, [0006] (emphasis added). That is, Halperin describes known antivirus software as creating a "mask that defines acceptable behaviour" (*i.e.*, legitimate software behavior), then monitoring the object and classifying as malware a program "if the actual monitored behaviour extends beyond that permitted by the mask" (*i.e.*, it exhibits any deviation from such behavior). As long as anomaly detection does not raise excessive false positives, it is a useful tool that would have been known to a POSITA and could have been expected to improve detection of previously unknown malware.

3. Analysis

71. It is my view that Halperin in view of Mankins renders claims 1, 2, 12-14, and 24 of the ’250 Patent obvious.

a. Independent Claim 1

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

72. To the extent this preamble is limiting, Halperin discloses [1P].

73. Halperin discloses “computer and computer network security in general, and more particularly to detection and prevention of malicious computer programs.” Ex. 1006, [0002]. Further, “[t]he proposed system works in two phases to identify malicious code” *id*. [0087] clearly discloses [1P]. Halperin also discloses that “when malicious software has been identified . . . it is possible to isolate the mechanism that has been spreading the malicious software,” *id.*, [0073]. A POSITA would have known that Halperin’s methods for detecting and identifying computer objects as malicious disclosed a “method of classifying a computer object as malware.”

[1A]: 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;

74. Halperin discloses [1A].

75. In Figure 5, Halperin discloses a base computer (server 502) and a first remote computer (one of computers 500). Ex. 1006, Fig. 5, [0047]. It also discloses that a base computer (e.g., server 502) receives data about a computer object from a first remote computer (one of computers 500); for example, it may receive "target behavior information" (Fig. 5) associated with a potential "virus." Ex. 1006, Fig. 5, [0036]-[0037], [0047].

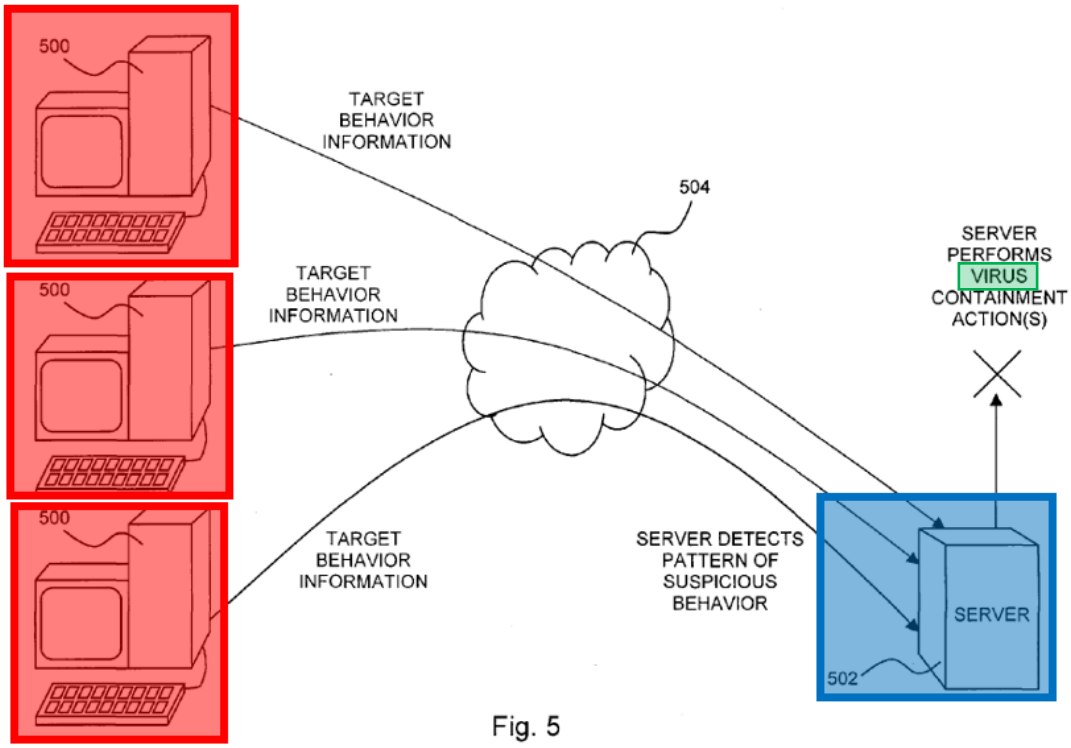

Fig. 5

76. In Halperin, "target behavior profiles are defined for **computers 500**. Each target behavior profile describes behavior that should be the subject of correlation analysis." *Id.*, [0038]. Therefore, **computers 500** can "detect associated target behavior and notify **server 502** accordingly" and **server 502** "collect[s]

information regarding target behavior detected at two or more of **computers 500**." *Id.*, [0047]. And Halperin discloses, with reference to embodiments including the Figure 5 embodiment, that "each computer" logs "any predefined suspicious event that takes place" and discloses "[t]hat information is logged in a central database," i.e., at the base computer. Ex. 1006, [0086].

77. A POSITA would have known that "target behavior" and "predefined suspicious event[s]" are data about a computer object like a **virus**, or similar computer objects stored on at least one of **computers 500**. After all, Halperin's purpose is to determine whether "a computer virus may have infected those computers." *Id.* [0047]. Therefore, Halperin discloses "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."

[1B]: determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware;

78. Halperin discloses [1B].

79. The method in Halperin "collect[s] information regarding target behavior detected at two or more of computers 500," and "determine[s] whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those

computers." Ex. 1006, [0047]. Halperin therefore discloses "determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware."

[1C]: classifying the computer object as malware when the data indicates that the computer object is malware;

80. Halperin discloses [1C].

81. As explained above, the method in Halperin compares the suspect object to predefined criteria that are characteristics of malicious software: If "the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior," the Halperin system classifies the computer object as malware (e.g., a "computer virus"). *See* Ex. 1006, [0047], [0057], [0073], [0074]; *id.* [0087] ("The proposed system works in two phases to identify malicious code (from here on referred to as 'viruses.')"); *id.*, [0061] ("[T]he identification of virus activity may include automatic identification of suspicious activity by a server or a combination of automatic identification and a notification of a system operator and approval by that operator that the suspicious activity is truly a virus, before notifying other servers.").

82. Further, a POSITA would have known that when the Halperin system responds to an object it has classified as a virus (e.g., by "activat[ing] a virus alarm" or "initiat[ing] virus containment actions," Ex. 1006, [0090]), it does so *because* the

system has classified the object as malware. Halperin therefore discloses "classifying the computer object as malware when the data indicates that the computer object is malware."

[1D]: when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware;

83. Halperin and Mankins disclose [1D].

84. As explained above, after Halperin's system "collect[s] information regarding target behavior detected at two or more of computers 500," and subsequently determines that the correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers," the Halperin system classifies the computer object as malware (e.g., a "computer virus"). *See* Ex. 1006, [0047], [0057], [0073], [0074]; *id.* [0087] ("The proposed system works in two phases to identify malicious code (from here on referred to as 'viruses.')"); *id.*, [0061] ("[T]he identification of virus activity may include automatic identification of suspicious activity by a server or a combination of automatic identification and a notification of a system operator and approval by that operator that the suspicious activity is truly a virus, before notifying other servers."). Further, a POSITA would have understood that the Halperin system's "activat[ion] of a virus alarm" and initiation of "virus

containment actions" would have occurred upon classification of an object as malware (e.g., a "virus"). *See id.* [0090].

85. It would have been obvious to a POSITA that when Halperin's determining *does not* indicate that the computer object is malware, that the Halperin system would do the opposite: no virus containment actions would be initiated, because the computer object has been classified as "not malware." Therefore, Halperin discloses "when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware."

86. Mankins also discloses this limitation. In Mankins, "normal (i.e., expected) system-call behavior is defined by monitoring system-call behavior during operation of the computer program under training conditions (e.g., when the computer program is not subject to intrusion attacks). Ex. 1008 at 3:37-40. For example, system-call behavior can be monitored "at the OS kernel." *Id.* at 6:63-7:7. The Mankins system allows programs to run as long as "a match exists" for the program's behavior in the Bloom filter data object representing "normal" behavior for that program. *Id.* at 11:30-37, 11:56-12:14, Fig. 3. A POSITA would have understood that Mankins's disclosure of running a computer program to obtain data on normal behavior, and then allowing the program to run so long as its behavior matches the predefined normal behavior means that, by default, the computer object is initially classified "as not malware" as required by this claim limitation.

Therefore, Mankins also discloses "when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware."

[1E]: 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;

87. Mankins discloses or renders obvious [1E].

88. Halperin discloses "target behavior profiles" of acceptable and/or unacceptable behavior that are "defined." *See, e.sg.*, Ex. 1006, [0047] ("[O]ne or more target behavior profiles are defined for computers 500.").

89. Mankins "uses a Bloom filter to generate a Bloom filter data object containing data representative of normal system-call behavior" of a computer program. Ex. 1008 at 3:40-43. In Mankins, the "predefined Bloom filter data is representative of 'normal' computer program behavior that is expected during operation of the computer programs when not subject to intrusion attempts." *Id.* at 3:26-30. The "Bloom filter data object 180 is built from the monitored behavior" of the computer object. *Id.* at 11:30-31. A POSITA would have known that Mankins's "Bloom filter data object," which is "representative of 'normal' computer program behavior" (Ex. 1008 at 3:26-30) is a "mask . . . that defines acceptable behaviour for the computer object."

90. Mankins also discloses that "Bloom filter data objects" are automatically generated: "The intrusion detection system 100 may be configured to automatically generate Bloom filter data objects 180 for computer programs 130 and then automatically use the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time." Ex. 1008 at 8:8-13; *see also id.* at 11:24-45 ("In some embodiments, steps 210 through 230 may be performed automatically to generate the Bloom filter data object 180 in an automated fashion, without requiring user intervention.").

91. Mankins discloses an exemplary configuration of its intrusion detection system implemented on a single computer 110 at the operating system kernel level. *See* Ex. 1008 at 6:58-8:13. In this exemplary configuration, the Bloom filter data objects are automatically generated and subsequently used to monitor system-call behavior at the single computer 110. Accordingly, it would have been obvious to a POSITA to automatically generate Bloom filter data objects at Halperin's remote computers 500, in the combined Halperin and Mankins system.

92. However, Mankins also explains that its "intrusion detection system 100 may be implemented *in a distributed fashion over a network of computing devices*." Ex. 1008 at 4:29-30 (emphasis added); *see also id.* at 4:50-52 ("In many embodiments, the computer 110 is in the form of one or more network servers." (emphasis added)). Thus, it would have also been obvious to a POSITA, in light of

these statements in Mankins, to implement Mankins's intrusion detection system in a distributed fashion, including through the use of Halperin's system architecture, by automatically generating Bloom filter data objects at the "server 502," or by collecting Bloom filter data objects generated by each computer 500 at the server 502 to obtain a more robust data set of "normal" program behavior, which would reduce false positives. *See* Ex. 1006, [0047], Fig. 5.

93. Mankins also discloses that "the mask" (i.e., the Bloom filter data object) "is generated in accordance with normal behaviour of the object determined from said received data" as required by this limitation. Mankins's Figure 2 is a flowchart that illustrates an example process for defining "normal system-call behavior" (*i.e.*, "normal" or "acceptable" behavior) for a computer program (*i.e.*, a "computer object") under training conditions:

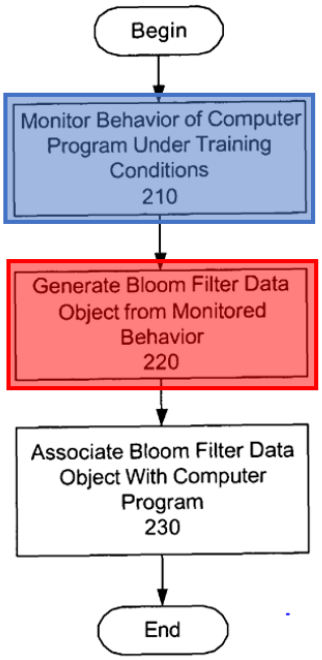

FIG. 2

Ex. 1006 at 11:19-23; Fig. 2. Mankins describes "monitoring behavior of a computer program under training conditions at **step 210**," which may be performed by "monitoring system-call behavior at the OS kernel 120 when the computer program 130 is not subject to intrusion attacks." *Id.* at 11:24-29. At **step 220**, "a profile Bloom filter data object 180 is built from the monitored behavior," which can "contain data representative of predefined normal behavior" that "is defined in the form of . . . sequences of system calls." *Id.* at 11:30-36. The Bloom filter data object is then "associated with the computer program." *Id.* at 11:37-38. It would have obvious to a POSITA that Mankins's description of "monitoring behavior of a computer program under training conditions" and building a Bloom filter data object

from that monitored behavior means that the Bloom filter data object (i.e., "the mask") "is generated in accordance with normal behaviour of the object determined from said received data." Therefore, Mankins discloses or 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 data."

94. As discussed above in Section VII.C.2, it would have been obvious to a POSITA to incorporate the Bloom filter mask of Mankins into the detection system of Halperin, and a POSITA would have been motivated to do so. Halperin's behavioral approach to virus detection determines that a computer object is malware if the "correlated target behavior corresponds to a predefined suspicious behavior pattern of target behavior." Ex. 1006, [0047]. In other words, Halperin describes behavior-based analysis wherein malware may be positively identified by predefined specific sequences of suspicious actions. However, Halperin also discloses the use of anomaly detection, an existing technique (and the technique used in Mankins) that "characteriz[es] legitimate software behavior and then identif[ies] any deviation from such behavior." *Id.*, [0006]. Accordingly, the substitution or addition of Halperin's predefined suspicious behavior pattern with Mankins's Bloom filter mask, which can determine an object is malware if that computer object performs actions beyond its "normal (i.e. expected) system-call behavior" (i.e., anomaly

detection – *see* Ex. 1008 at 3:37), would have been a simple substitution (or addition) of one known behavior-based approach to malware detection for another.

95. Moreover, a POSITA would have recognized that as long as anomaly detection does not raise excessive false positives, it is a useful tool that would have been known to a POSITA and could have been expected to improve detection of previously unknown malware. Thus, the use of Mankins's Bloom filter data object ("mask") approach in a distributed system such as Halperin would actually improve the system by reducing positives by improving the robustness of the "normal" behavior data represented in the Bloom filter data objects.

96. Halperin explains that because "computer user behavior is quite dynamic and tends to vary over time and between different users," the "application of heuristic techniques thus often results in a false alarm." Ex. 1006, [0006]. In my view, that is why Halperin discloses a distributed approach using information collected from multiple remote computers. Mankins also notes that a distributed approach can correct for false positives: "In particular, when several network devices generate separate Bloom filter data objects 180, are monitored together, and are subject only to attacks that affect each of the network devices within a small amount of time, it is likely that at least one of the network devices would detect a false positive that is produced by another device." Ex. 1008 at 10:53-62. Thus, a POSITA would have understood that implementation of Mankins's Bloom filter mask in a

distributed system like that of Halperin would have reduced the occurrence of false positives because the Bloom filter data objects would be generated from behavioral data collected from different computers having different users, and at different times, under differing operating conditions and use patterns.

97. A POSITA would have also had a reasonable expectation of success in modifying Halperin to incorporate Mankins's Bloom filter data object anomaly detection approach. As discussed above, anomaly detection was a known approach to virus detection. Thus, adding Mankins's particular technique using Bloom filter data objects would have been nothing more than the use of a known technique to improve a similar device (Halperin's intrusion detection system) in a similar way.

[1F]: running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers;

98. Mankins discloses [1F].

99. Mankins's Figure 3 is a flowchart that illustrates "an example process for detecting intrusion attempts by monitoring system-call behavior":

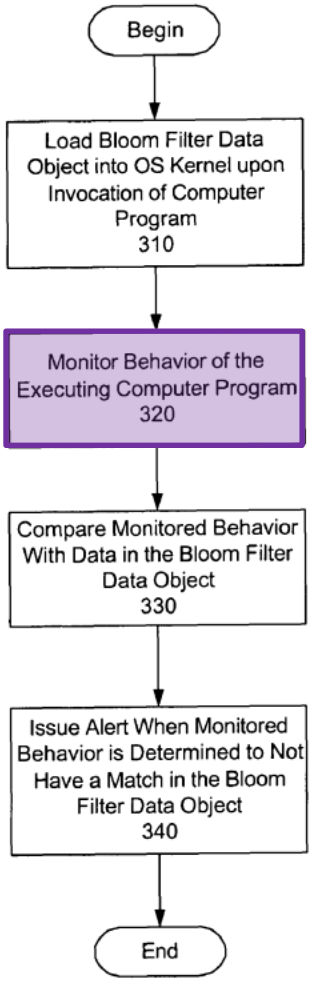

FIG. 3

Ex. 1008 at 11:50-52, Fig. 3. At step 320 of Figure 3, "behavior of the computer program is monitored as it is executed by the computer 110." *Id.*, 11:63-65. Mankins teaches "automatically us[ing] the Bloom filter data objects 180 to monitor system-call behavior in the OS kernel 120, thereby allowing intrusions to be detected in real time." *Id.* at 8:8-13. Further, Mankins discloses that the automatically monitored behavior may be "system-call behavior at the OS kernel." *Id.* at 11:65–66; *see also id.* at 6:3–56 (describing "the monitoring subsystem 140" that may "use

[a] rolling log of system calls to monitor system-call sequences occurring in the OS kernel"). A program can only make system calls to the OS if it is running. Thus, a POSITA would have found it obvious that Mankins's disclosure of "monitoring system-call behavior" for a program to detect intrusions "in real time" meant that the monitored programs were "running," as required by this claim limitation. *See id.* at 11:46-50 ("[T]he intrusion detection system 100 is prepared to detect intrusion attempts by monitoring the behavior of the computer program 130 *when it is executed*." (emphasis added)).

100. As explained above, Mankins discloses an embodiment wherein "several network devices generate separate Bloom filter data objects 180, [and] are monitored together." Ex. 1006 at 10:57-58. A POSITA would have found it obvious that where network devices are "monitored together," any program whose system call sequences are monitored would likewise be "running" as required by [1F]. In Mankins's networked/distributed embodiment, each of "several network devices [i.e., plural remote computers] generate separate Bloom filter data objects 180." Ex. 1008, 10:57–58. Mankins indicates that the purpose of this arrangement is to "reduce[]" "the size of the Bloom filter data object 180." *Id.*, 10:53-56. This reduction in size occurs because each remote computer has its own Bloom filter data object. Further, this arrangement requires the program be run on the remote computers, where its behavior will be compared to their computer's Bloom filter

data object for that program. Thus, Mankins discloses "running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers."

101. As discussed above in Section VII.C.2, a POSITA would have found it obvious incorporate Mankins's Bloom filter data objects into the architecture of Halperin, and to compare the monitored system call sequences of an executing program either directly at a remote computer (i.e., Halperin's "computers 500") or at a base computer (i.e., Halperin's "server 502"). In other words, the combined Halperin/Mankins system could log the sequence of system calls made by the object in the kernel (at each computer 500) and maintain the Bloom filter at each remote computer, as in the preferred embodiment in Mankins. Conversely, it would have also been obvious to a POSITA to have collected system call sequences at the remote "computers 500," and transmitted the sequences as part of the data that is received at the "server 502," and to have compared them against a Bloom filter data object stored at the server. As explained above, a POSITA would have been motivated to use this distributed approach in order to reduce false positives by utilizing a more robust data set to generate the Bloom filter data objects representing "normal" behavior at the server 502. Because either approach could be implemented successfully, this choice between local and server-based comparison of system call

sequences would have been nothing more than an obvious and routine matter of system design to a POSITA.

[1G]: allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask;

102. Mankins discloses element [1G].

103. At step 330 of Mankins's Figure 3, "the monitored behavior is compared with data in the Bloom filter data object," which "may be performed by the monitoring subsystem 150, which may use the Bloom filter 175 to search the Bloom filter data object 180 to determine whether a match exists for any particular set of monitored behavior (e.g., a system-call sequence)." Ex. 1008, 12:1-6.

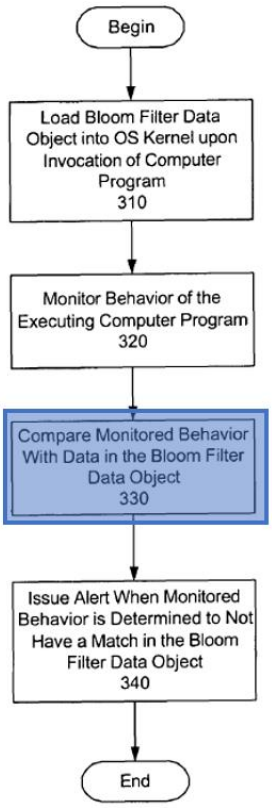

FIG. 3

Id., Fig. 3. Mankins explains that, as with conventional anomaly-detection techniques, ""[a]s long as the monitored system-call sequences have a match" in the Bloom filter data object, the "operation is considered to be normal" and the object or program will be permitted to continue to run. *Id.* at 1:60-63, 2:8-10. On the other hand, "[i]f a match is not found, the observed system-call sequence is determined to be anomalous and may indicate an intrusion attempt." *Id.* at 6:39-41. Thus, Mankins discloses "allowing the computer object to continue to run when behaviour of the computer object is permitted by the mask."

104. As explained above in Section VII.C.2, a POSITA would have found it obvious incorporate Mankins's Bloom filter data objects into the architecture of Halperin, and would have been motivated to do so. Both Halperin and Mankins disclose techniques for behavior-based approaches to detecting previously-unknown malware, and a POSITA would have been motivated to pursue an "integrated" or "combined" approach to malware detection, such that if one approach fails (e.g., Halperin's system does not detect predefined suspicious behavior), then another may succeed (e.g., Mankins's system detects abnormal system call sequences that do not necessarily meet Halperin's predefined suspicious behavior). Moreover, a POSITA would have had a reasonable expectation of success in incorporating Mankins's Bloom filter data object techniques in Halperin, because Halperin's system already performs a similar behavior-based malware detection approach, and Halperin

explicitly acknowledges the use of anomaly-based malware detection like that disclosed in Mankins. *See* Ex. 1006, [0006] ("Another approach involves using antivirus software that employs heuristic techniques to identify typical virus behavior by characterizing legitimate software behavior and then identifying any deviation from such behavior.").

105. Accordingly, the Halperin system, while utilizing Mankins's Bloom filter technique, would have allowed a computer object "to continue to run when behaviour of the computer object is permitted by the mask," as required by [1G].

[1H]: disallowing the computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask; and,

106. Mankins discloses [1H].

107. Mankins explains "[i]f a match [between the Bloom filter data object 180 and the actions of the computer object] is not found, the observed system-call sequence is determined to be anomalous and may indicate an intrusion attempt." Ex. 1008 at 6:39-41. In step 340 of Mankins's Figure 3, "an alert is issued in response to a determination that the Bloom filter data object 180 does not contain a match for the monitor behavior," and "[a]ny appropriate predefined action may be performed in response to the alert. *Id.* at 12:7–14. For example, "the report subsystem 170 may be configured to automatically invoke predetermined actions designed to protect the computer." *Id.* at 11:8-11. Accordingly, Mankins discloses "disallowing the

computer object to run when the actual monitored behaviour of the computer object extends beyond that permitted by the mask."

108. A POSITA would have found it obvious to disallow a program that performed a system-call sequence not found in the Bloom filter data object from running in the combined Halperin/Mankins system. As explained with respect to [1C] above, Halperin discloses that if suspicious behavior is detected, that its system can "activate a virus alarm" or "initiate virus containment actions." Ex. 1006, [0090]. It would have been obvious to a POSITA that in the combined Halperin/Mankins system, that appropriate "virus containment actions" would have included Mankins's disclosure of "predetermined actions designed to protect the computer," which would include disallowing a program from running when its behavior does not match that in the Bloom filter data object.

[1I]: reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

109. Limitation [1I] is rendered obvious by Mankins.

110. As explained above regarding [1G], Mankins teaches that after the comparison of the monitored system-call behavior with data in the Bloom filter data object, "[i]f a match is not found, the observed system-call sequence is determined to be anomalous and may indicate an intrusion attempt." Ex. 1008 at 6:39-41.

Subsequently, the system may issue an alert, and take "predetermined actions designed to protect the computer." *Id.* at 11:8-11.

111. It would have been obvious to a POSITA that taking "predetermined actions to protect the computer" would have included reclassifying the program that performed the system-call sequence that was flagged as malware for having no match in the Bloom filter data object as malware. In the combined Halperin/Mankins system, the system-call sequence that was flagged could be added as a "predefined suspicious behavior" pattern for use with Halperin's initial intrusion detection system. Alternatively, a signature could be generated for the program that generated the flagged system-call sequence and used with a conventional signature-based virus detection approach for known malware. Thus, Mankins renders obvious "reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask."

112. As I explained with respect to [1C] above, Halperin discloses that if suspicious behavior is detected, that its system can "activate a virus alarm" or "initiate virus containment actions." Ex. 1006, [0090]. It would have also been obvious to a POSITA that in the combined Halperin/Mankins system, that taking appropriate "virus containment actions" as described in Halperin would have included reclassifying a program as malware.

b. Independent Claim 13

113. Claim 13 includes the same claim limitations as Claim 1, except that it further claims an apparatus comprising "a base computer constructed and arranged to" perform the method claimed by Claim 1. Halperin discloses a base computer (server 502), as discussed above, that, in view of Mankins, does perform all of the steps of Claim 1. Therefore, Halperin and Mankins also teach an apparatus "constructed and arranged to" perform the functions in Claim 13. Therefore, just as Claim 1 is obvious over Halperin in view of Mankins, Claim 13 is obvious over Halperin in view of Mankins.

c. Dependent Claims 2 and 14

114. Claim 2 depends from Claim 1 and further requires "wherein the data about the computer object that is sent from the plural remote computers to the base computer includes one or more of: . . . . the current name of the computer object." Halperin discloses this element too.

115. In one of Halperin's embodiments, the system detects malware by identifying "computers in the network sending the same file attachment via a message in the last 15 minutes." Ex. 1006, [0054]. A POSITA would have known that a common, well known way to determine whether two files are the same is to compare their names. And a POSITA would have understood that the name of the computer object is part of the data sent by the remote client computer to the server

502 (base computer); that objects with the same name are at least related in this respect; and that it was common to check the creation and modification dates of an object to determine whether an apparently safe object has been changed. In addition, it was common and well known in the art at the time to check the creation and modification dates of an object to determine whether a nominally safe application may have been changed. Accordingly, Claim 14 is obvious over the combination of Halperin and Mankins.

116. Claim 14 is essentially identical to Claim 2: Claim 14 is the apparatus version of Claim 2. Because Halperin and Mankins render Claim 2 obvious, the combination therefore renders Claim 14 obvious as well.

d. <u>Dependent Claims 12 and 24</u>

117. Claim 12 depends from Claim 1 and further requires "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." Mankins discloses this additional limitation.

118. Mankins teaches that upon "detection of anomalous program behavior," (i.e., "non-permitted behaviour that extends beyond that permitted by the

mask"), its report subsystem may issue "an alert and allow execution of detected anomalous behavior to proceed." Ex. 1008 at 11:4-7.

119. One way of "allow[ing] execution of detected anomalous behavior to proceed" would have been to compare the system-call sequence flagged as anomalous with the Bloom filter data objects for *other* instances of the same program, and only allowing further execution if the sequence found a match within the "normal" behavior of one or more of the other instances. In fact, Mankins touts this as an advantage of the distributed approach: "[W]hen several network devices generate separate Bloom filter data objects 180, are monitored together, and are subject only to attacks that affect each of the network devices within a small amount of time, it is likely that at least one of the network devices would detect a false positive that is produced by another device. Thus, the sizes of the Bloom filter data objects 180 may be reduced because the different network devices can work together to identify false positives. *Id*. at 10:53-64. A POSITA would have found it obvious to subsequently add the system-call sequence initially flagged as anomalous to the Bloom filter data object for the process that generated the system-call sequence, in order to prevent future executions of that particular system-call sequence from being flagged as anomalous. Further, it would have been obvious to a POSITA to perform this functionality at Halperin's server 502, since as discussed regarding [1A], the server 502 (i.e., the "base computer") receives data about objects from the computers

500 (i.e., the "remote computers"). Accordingly, Claim 12 is obvious over the combination of Halperin and Mankins.

120. Claim 24 is essentially identical to Claim 12: Claim 24 is the apparatus version of Claim 2. Because Halperin and Mankins render Claim 12 obvious, the combination therefore renders Claim 24 obvious as well.

D. Ground 2: Halperin in View of Kirby and Mankins Renders Obvious Claims 1-30

1. Prior Art Overview

121. Yet another prior-art reference, Kirby, also discloses many limitations of the claims of the '250 Patent. Like Halperin, Kirby discloses a method and apparatus for virus detection. Ex. 1007, abstract. Kirby uses two processes in parallel: a signature-matching approach and a behavior-monitoring approach. *Id.* at 2:56-62. The latter technique, the behavior-monitoring approach, determines whether the actions taken by suspected malware are characteristic of the types of actions of actions often taken by malware. *Id.* at 10:54-60 (Kirby can "determine if the detected actions correspond to the suspicious behavioral actions of the behavior heuristic 112"). If an object takes such actions, the system in Kirby can designate that object as malware, *id.* at 10:63-11:6; *see also id.* at 15:7-20. Kirby can also create a "hash signature" of an object at the computer on which the object is found,

"therefore enabling a signature based (immediate) prevention mechanism against subsequent occurrences of the malware on this or other computers." *Id.* at 15:15-20.

2. Motivation to Combine and Reasonable Expectation of Success

122. Kirby, like Halperin, discloses malware-detection systems and processes that monitor the behavior of computer objects as a way of detecting malware that has not been previously seen. Ex. 1006, [0005], [0007]; Ex. 1007 at 1:62-2:7, 2:12-25 ("Embodiments of the invention provide a behavioral approach to virus detection and processing of other malicious processing activity, that in contrast to conventional techniques, exhibits a substantial departure from the conventional day Zero problem."). In fact, Kirby uses both a behavior-based approach and a signature-matching approach. Ex. 1007 at 2:56-62. As explained above, Mankins focuses on the use of Bloom filter data objects that contain data that is representative of "normal" and expected program behavior. Ex. 1008 at 3:40-43. The Mankins approach identifies anomalous behavior "when the actual system-call behavior does not have a match in the Bloom filter data object." *Id.* at 3:53-55.

123. A POSITA would have been motivated to combine Halperin, Mankins, and Kirby—among other reasons, doing so would produce the obvious, beneficial, and predictable result of incorporating Kirby's signature-matching to aid in the classification of malware. It would have been obvious to a POSITA to use

Kirby in this way, since Kirby's bimodal system of behavior-analysis and signature-matching to classify malware was well known as a way to detect malware that was effective and faster than a traditional single-mode approach.

124. A POSITA would also have been motivated to implement Kirby's disclosure of identifying objects that "touch" malware as malware themselves. *See, e.g.*, Ex. 1007 at 5:48-50 ("[A]ny process or file which 'touches' any untrusted file may 'inherit' an untrustworthy trait, which may then be passed on until the pattern indicates a correspondence with undesirable, and hence, preventable, behavior."). In Halperin, viruses are "designed to infiltrate computer files and other sensitive areas on the computer." Ex. 1006, [0003]. This means that viruses can modify programs or other files by changing them to include viral code. A POSITA would have understood that, because viruses modify objects to include viral code, an object that has been modified by a virus would be likely to be malware itself. Therefore, a POSITA would have been motivated to incorporate into Halperin Kirby's teachings regarding classifying objects that "touch" malware as malware themselves.

125. A POSITA would also have been familiar with information-flow-based integrity policy (from Biba). Such policy dictates that if information flows from malware to another, higher-integrity object (e.g., by reading or writing), the object that was previously higher-integrity can now have no higher integrity than the low-integrity object (e.g., malware) that it came from. A POSITA would have

known that the same applies to processes—if a process, for instance, reads from a low-integrity or contaminated object, it must also be classified as low integrity.

3. Analysis

a. Independent Claim 1

126. As explained above, in my view Halperin in view of Mankins is sufficient to render obvious independent Claim 1 of the ’250 Patent. Claim 1 is also rendered obvious by the combination of Halperin, Mankins, and Kirby, for the additional reasons outlined below.

[1D]: when the determining does not indicate that the computer object is malware, initially classifying the computer object as not malware;

127. As explained above, in my view, both Halperin and Mankins disclose this limitation. Kirby also discloses [1D].

128. In Kirby, “[i]f the pattern matcher concludes that a content based check against the baseline of previously stored realtime signatures in the signature database indicates a *benign transmission*, then the behavior monitor evaluates the actions performed by, on behalf of or as a result of the incoming transmission, based on the operations exhibited.” Ex. 1007 at 7:65-8:4 (emphasis added). A POSITA would have known that determining an object is initially benign is classifying a new object as not malware. Thus, Kirby discloses “when the determining does not

indicate that the computer object is malware, initially classifying the computer object as not malware."

[1F]: running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers;

129. As explained above, in my view, Mankins discloses this limitation. Kirby also discloses [1F].

130. In Kirby, "the security agent 110 is configured to intercept and monitor a variety of subject (process or thread) actions, such as file accesses, I/O port accesses, and other security pertinent operations." Ex. 1007, 11:51-54. In order to "intercept and monitor" those actions, Kirby allows objects that have not yet been identified as malware to run on its host computers (which include the "security agent"). Thus, it would have been obvious to a POSITA that Kirby's "intercept and monitor" operation is the same as "running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers." Thus, Kirby discloses "running said object on at least one of the remote computers; automatically monitoring operation of the object on the at least one of the remote computers."

[1I]: reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.

131. As explained above, in my view, Mankins discloses this limitation. Kirby also discloses [1I].

132. Kirby classifies a computer object as benign (*i.e.*, as not malware) if the object passes the "signature check, which generally occurs immediately upon introduction of a new entity (e.g. file or object)." Ex. 1007 at 14:30-33. The Kirby system also continues "the behavioral monitoring and prevention, which occurs as an ongoing monitoring effort." Ex. 1007 at 14:35-38. Thus the second step ("behavioral monitoring and prevention") occurs after the computer object is classified as benign (*i.e.*, as not malware). And if a POSITA were to deploy Mankins's Bloom filter data object within Kirby, it too would monitor a computer object after the object passed the signature step (and was classified as benign). Therefore, if the behavior-monitoring step determines that the computer object is malware, it must *re*classify that computer object as malware. Accordingly, Kirby discloses "reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask."

b. <u>Independent Claim 13</u>

133. Claim 13 includes the same claim limitations as Claim 1, except that it further claims an apparatus comprising "a base computer constructed and arranged

to" perform the method claimed by Claim 1. Halperin discloses a base computer (server 502), as discussed above, that, in view of Mankins and in further view of Kirby, does perform all of the steps of Claim 1. Therefore, Halperin, Mankins, and Kirby also teach an apparatus "constructed and arranged to" perform the functions in Claim 13. Therefore, just as Claim 1 is obvious over Halperin in view of Mankins and in further view of Kirby, Claim 13 is obvious over Halperin in view of Mankins and in further view of Kirby.

c. <u>Dependent Claims 2 and 14</u>

134. As discussed above, Claims 2 and 14 are obvious over Halperin in view of Mankins. For the same reasons, Claims 2 and 14 are also obvious over Halperin in view of Mankins and in further view of Kirby.

d. <u>Dependent Claims 3 and 15</u>

135. Claim 3 depends from Claim 1 and further requires "wherein the determining identifies relationships between the object and other objects." Kirby discloses this additional limitation.

136. In Kirby, a relationship may be found between one object and other objects. "For example, an incoming email attachment file may be deemed as 'untrusted.' A process reading the file and writing another file creates another untrusted file . . . any process or file which 'touches' an untrusted file may 'inherit' an untrustworthy trait" Ex. 1007 at 5:48-53. Also, the Kirby system may monitor

those related objects together and can identify a common object of the related objects as malware when "the pattern indicates a correspondence with undesirable . . . behavior." Ex. 1007, at 5:54-55. In other words, Kirby's disclosure of the inheritance of an untrustworthy trait from one process or file that "touches" an untrusted file is an example of determining that a process or file is *related to* a malicious process or file. A POSITA would have known through these disclosures that Kirby discloses determining that a computer object is malware by identifying relationships between the object and other objects. Accordingly, Claim 3 is obvious over the combination of Halperin, Mankins, and Kirby.

137. Claim 15 is essentially identical to Claim 3: Claim 15 is the apparatus version of Claim 3. Because Halperin, Mankins, and Kirby render Claim 3 obvious, the combination therefore renders Claim 15 obvious as well.

e. <u>Dependent Claims 4 and 16</u>

138. Claim 4 depends from Claim 3 and further requires "wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware." Kirby discloses this additional limitation.

139. In Kirby, a signature that is computed from a malware object "is deemed operable to indicate subsequent undesirable activity," for example, by other objects that perform the same suspicious behavioral actions. Ex. 1007 at 10:61-11:6. Kirby explains that "the same *or similar object appearing elsewhere* is likely

indicative of the same virus or undesirable transmission." *Id.* at 12:27-30 (emphasis added). And, as explained with respect to Claim 3 above, Kirby teaches that "any process or file which 'touches' an untrusted file may 'inherit' an untrustworthy trait." *Id.* at 5:52-53.

140. It would have been obvious to a POSITA that Kirby's disclosure of "similar object[s] appearing elsewhere" would include those objects Kirby describes that "touch" an untrusted file, such as a child (a file created by the untrusted file) or parent (the file that created the untrusted file). Thus, the signature that is computed from a malware object would serve to classify "the same or similar object[s] appearing elsewhere" as malware, based on the previous classification. Accordingly, Claim 5 is obvious over the combination of Halperin, Mankins, and Kirby.

141. Claim 16 is essentially identical to Claim 4: Claim 16 is the apparatus version of Claim 4. Because Halperin, Mankins, and Kirby render Claim 4 obvious, the combination therefore renders Claim 16 obvious as well.

f. <u>Dependent Claims 5 and 17</u>

142. Claim 5 depends from Claim 3 and further requires "wherein said other objects include the object or similar objects stored on at least some of the remote computers." Halperin and Kirby each disclose this additional limitation.

143. Halperin discloses this ability: "A certain percentage of the computers in the network sending file attachments via one or more messages in the last hour where each of the files includes the same string of bits" is an example of a predefined pattern of suspicious behavior. Ex. 1006, [0055] (emphasis added); *see also id.*, [0047]. A POSITA would have known that "the same string of bits" is, or could be, the virus, malware, or attached file containing the string itself. And since the same embodiment can have the virus or malware located on multiple computers, Halperin therefore discloses this element.

144. As explained above regarding Claim 3, Kirby teaches that one way to identify potentially malicious code is by determining that code is related to malicious code found on at least some of the remote computers. Ex. 1007 at 5:49-50. Likewise, Kirby explains that "the same *or similar object appearing elsewhere* is likely indicative of the same virus or undesirable transmission." *Id.* at 12:27-30 (emphasis added). It would have therefore been obvious to a POSITA that Kirby's disclosure of "the same or similar object[s] appearing elsewhere" meets this additional limitation of Claim 5. Accordingly, Claim 5 is obvious over the combination of Halperin, Mankins, and Kirby.

145. Claim 17 is essentially identical to Claim 5: Claim 17 is the apparatus version of Claim 5. Because Halperin, Mankins, and Kirby render Claim 5 obvious, the combination therefore renders Claim 17 obvious as well.

g. Dependent Claims 6 and 18

146. Claim 6 depends from Claim 3 and further requires "wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object." Kirby discloses this additional limitation.

147. In Kirby, "an incoming email attachment file may be deemed as 'untrusted.' A process reading the file and writing another file creates another untrusted file. A subsequent attempt to execute the file may constitute malicious behavior." Ex. 1007, 5:48–52. Therefore, a computer object classified as "untrusted" (i.e., as malware) will create "another file" (i.e., a child object), which Kirby will also classify as "untrusted" (i.e., as malware).

148. And "the invention" in Kirby "focuses on the subject actor of suspect operations, such as the process writing or executing a file…. the behavioral approach analyzes the actions performed and the provenance of the data used…." Ex. 1007, 2:25-30. A POSITA would have known that when one process engages in the action of "executing a file," that process (an object) creates a child process object (which is executing the code in that other file). Therefore, it is clear that the invention in Kirby identifies an object that executes a malicious file (i.e., a parent object to malware).

149. Kirby also teaches that "the security agent 110 is configured to intercept and monitor a variety of subject (process or thread) actions, such as file accesses, I/O port accesses, and other security pertinent operations." Ex. 1007, 11:51-54. Kirby also states that "the suspicious actions are codified occurrences derived from common denominators of undesirable transmissions, such as viruses and worms." *Id.*, 11:41-44. Therefore, Kirby was aware of common worm behaviors, such as writing an executable file and then creating a child process that executes that malware file. "[T]he invention focuses on the subject actor of suspect operations, such as the process writing or executing a file…. the behavioral approach analyzes the actions performed and the provenance of the data used…." *Id.*, 2:25-30. When Kirby detects "undesirable behavior" and identifies the associated object (i.e., the file that was executed whose child process exhibited bad behavior), the system is concerned with the provenance of the malware and its method of conveyance, and therefore the parent process that spawned the detected malware child and the object containing the instructions it was executing. Accordingly, Claim 6 is obvious over the combination of Halperin, Mankins, and Kirby.

150. Claim 18 is essentially identical to Claim 6: Claim 18 is the apparatus version of Claim 6. Because Halperin, Mankins, and Kirby render Claim 6 obvious, the combination therefore renders Claim 18 obvious as well.

h. Dependent Claims 7 and 19

151. Claim 7 depends from Claim 1 and further requires "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." Kirby discloses this additional limitation.

152. The system in Kirby both scans for viruses and monitors object behavior. In Kirby, the host computer (*i.e.*, remote computer) has a "security agent" with "a scanner/extractor 124 for parsing objects such as files and delivering byte sequences to the signature generator 114 for computing a signature or hash." Ex. 1007, 11:11-16. And in Kirby, moreover, the remote computer generates a signature, which "is an identifier such as a hash, i.e. MD4, MD5, or SHA to name several known in the art, which are rapidly computable," and that "other hashes or computations which are smaller than the entity they represent yet highly unlikely to be duplicated by a dissimilar entity" could be used. Ex. 1007, 12:37-43.

153. In Kirby the remote computer will also calculate the hash for objects stored on the remote computers and then send the hash to the base computer, in order to give the base computer data about the computer objects. Ex. 1007 at 9:66-10:4, 15:21-26. A POSITA would have found it obvious to modify Halperin's disclosure of transmitting object data to a base computer, by combining it with Kirby's disclosure of transmitting data in the form of a key developed by hashing vis-à-vis

the objects on remote computers—doing so would have been an obvious combination or prior-art elements in according to known methods to yield predictable results. Accordingly, Claim 7 is obvious over the combination of Halperin, Mankins, and Kirby.

154. Claim 19 is essentially identical to Claim 7: Claim 19 is the apparatus version of Claim 7. Because Halperin, Mankins, and Kirby render Claim 7 obvious, the combination therefore renders Claim 19 obvious as well.

i. <u>Dependent Claims 8 and 20</u>

155. Claim 8 depends from Claim 7 and further requires "wherein the key has at least one component that represents executable instructions contained within or constituted by the object." Kirby discloses this additional limitation.

156. As I noted with regard to Claim 7, in the Kirby system, object data is sent to a base computer or AV server as a hash. And in Kirby, "the signature is computed from the associated object (e.g. infected email attachment) of the behavioral actions and operable to indicate subsequent undesirable activity." Ex. 1007 at 12:23-26. A POSITA would have known that Kirby's signature could be "operable to indicate subsequent undesirable activity" by including some component representing executable instructions in (or constituted by) the malware object. A POSITA would have found it obvious to include that component, and that doing so would have improved detection ability.

157. Moreover, Counsel for the Petitioners have informed that in its infringement contentions Patent Owner claims that sending a SHA256 hash meets this limitation. But Kirby discloses this element: It explicitly discloses sending a "SHA" hash. Ex. 1007 at 12:38-39. Accordingly, Claim 8 is obvious over the combination of Halperin, Mankins, and Kirby.

158. Claim 20 is essentially identical to Claim 8: Claim 20 is the apparatus version of Claim 8. Because Halperin, Mankins, and Kirby render Claim 8 obvious, the combination therefore renders Claim 20 obvious as well.

j. <u>Dependent Claims 9 and 21</u>

159. Claim 9 depends from Claim 7 and further requires "wherein the key has at least one component that represents data about said object." Kirby discloses this additional limitation.

160. As I noted above, the Kirby system generates a key "computed from the associated object (e.g. infected email attachment) of the behavioral actions," i.e., computed from the malware object. Ex. 1007 at 12:23-26. A POSITA would have known, therefore, that Kirby's key includes a component representing computer-object data. Kirby also explains that "the set of signatures may include signatures generated from predetermined mathematical computations on *at least a portion of the object*." *Id.* at 6:6-10 (emphasis added). A POSITA would have found it obvious that a signature that is generated from "mathematical computations on *at*

least a portion of the object" would necessarily mean that the signature would include data about the object as a component.

161. And, as I also noted above, a POSITA would have found it obvious to modify Halperin to include Kirby's key. Accordingly, Claim 9 is obvious over the combination of Halperin, Mankins, and Kirby.

162. Claim 21 is essentially identical to Claim 9: Claim 21 is the apparatus version of Claim 9. Because Halperin, Mankins, and Kirby render Claim 9 obvious, the combination therefore renders Claim 21 obvious as well.

k. <u>Dependent Claims 10 and 22</u>

163. Claim 10 depends from Claim 9 and further requires:

> 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.

Kirby discloses this additional limitation.

164. As I noted above in the discussion of Claim 9, Kirby's key is "computed from" the computer object. A POSITA would have understood that a key could include any of the information enumerated in Claim 11 if that key was computed from the computer object. A POSITA would have known to use at least one of the items enumerated in Claim 11 to improve the system's detection ability. Therefore, just as I discussed above regarding Claim 9, the combination of Halperin, Mankins, and Kirby also renders Claim 10 obvious.

165. Claim 22 is essentially identical to Claim 10: Claim 22 is the apparatus version of Claim 10. Because Halperin, Mankins, and Kirby render Claim 10 obvious, the combination therefore renders Claim 22 obvious as well.

l. Dependent Claims 11 and 23

166. Claim 11 depends from Claim 7 and further requires "wherein the key has at least one component that represents the physical size of the object." Kirby discloses this additional limitation.

167. As I have discussed, in Kirby a key is generated and it is "computed from" the computer object. A POSITA would have understood that a key that has been computed from a computer object could include an element representing the physical size of that object, and a POSITA would have found it obvious to include such information to improve detection.

168. Specifically, in Kirby, the remote computer generates a signature, which "is an identifier such as a hash, i.e. MD4, MD5, or SHA to name several known in the art, which are rapidly computable," and that "other hashes or computations which are smaller than the entity they represent yet highly unlikely to be duplicated by a dissimilar entity" could be used. Ex. 1007 at 12:37-43.

169. As I discussed respecting Claims 9 and 21, the Kirby system generates a key "computed from the associated object (e.g. infected email attachment) of the behavioral actions," i.e., computed from the malware object. *See id.* at 12:23-26. A POSITA would have understood that object metadata could help determine whether it was benign or malware. For instance, its size might be compared: An object's size might be compared to the known, expected characteristics of the object to determine whether it is likely malware. Therefore, a POSITA would have found it obvious to include at least one element representing the object's physical size in the key representing data about the object. Accordingly, Claim 11 is obvious over the combination of Halperin, Mankins, and Kirby.

170. Claim 23 is essentially identical to Claim 11: Claim 23 is the apparatus version of Claim 11. Because Halperin, Mankins, and Kirby render Claim 11 obvious, the combination therefore renders Claim 23 obvious as well.

m. Dependent Claims 12 and 24

171. As discussed above, Claims 12 and 24 are obvious over Halperin in view of Mankins. For the same reasons, Claims 12 and 24 are also obvious over Halperin in view of Mankins and in further view of Kirby.

n. Independent Claim 25

[25P]: 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:

172. *See* discussion of [1P] in Ground 1, above.

[25A]: providing from a remote computer to a base computer data about a computer object that is stored on the remote computer;

173. *See* discussion of [1A] in Ground 1, above.

[25B]: 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;

174. *See* discussion of [1A], and Claims 2 and 14 in Ground 1, above.

[25C]: 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,

175. *See* discussion of Claims 7 and 19, above.

[25D]: 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;

176. *See* discussion of [1E] in Ground 1, above.

[25E]: executing the computer object at the remote computer; automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;

177. *See* discussion of [1F], above.

[25F]: allowing the object to continue to run when behaviour of the computer object is permitted by the mask;

178. *See* discussion of [1G] in Ground 1, above.

[25G]: disallowing the object to run when the actual monitored behaviour of the object extends beyond that permitted by the mask; and,

179. *See* discussion of [1H] in Ground 1, above.

[25H]: reclassifying the object as malware when the actual monitored behaviour extends beyond [t]hat permitted by the mask,

180. *See* discussion of [1I], above.

[25I]: wherein the base computer is arranged to receive information about the behaviour of the object running on one or more remote computers,

181. *See* discussion of [1A] in Ground 1, above.

[25J]: 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.

182. *See* discussion of [1E] in Ground 1, above.

o. <u>Dependent Claim 26</u>

183. *See* discussion of Claims 8 and 20, above.

p. <u>Dependent Claim 27</u>

184. *See* discussion of Claims 9 and 21, above.

q. <u>Dependent Claim 28</u>

185. *See* discussion of Claims 10 and 22, above.

r. <u>Dependent Claim 29</u>

186. *See* discussion of Claims 11 and 23, above.

s. <u>Dependent Claim 30</u>

187. Claim 30 depends from Claim 25 and further requires "[a] non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25." Kirby discloses this additional limitation.

188. Kirby teaches that "[o]ne such embodiment comprises a computer program product that has a computer-readable medium including computer program logic encoded thereon that, when performed in a multiprocessing computerized device having a coupling of a memory and a processor, programs the processor to perform the operations disclosed herein as embodiments of the invention to carry out data access requests." Ex. 1007 at 6:48-55. It would have been obvious to a POSITA that this embodiment of Kirby represents "[a] non-transitory storage

medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25," which is obvious over the combination of Halperin, Mankins, and Kirby.

VIII. SECONDARY CONSIDERATIONS

189. I am not aware of any evidence of secondary considerations of non-obviousness relied upon by Webroot. To the extent Webroot is permitted to provide any evidence of secondary considerations to rebut my opinions expressed herein, I reserve the right to respond by supplementing and/or amending this declaration or submitting a reply declaration to consider the alleged secondary considerations of non-obviousness.

IX. RIGHT TO SUPPLEMENT

190. I reserve the right to supplement this declaration, if appropriate, should additional information come to light that is relevant to my opinions in this declaration. I also reserve the right to consider and comment on additional evidence, e.g., objective evidence of non-obviousness, that may be presented by Webroot or its experts.

191. At any trial or hearing at which I may testify in this matter, I may provide demonstrative aids, such as computer animations, excerpts, and figures from relevant exhibits and prior-art references, and physical examples to assist in explaining the subject matter in my report.

192. I will continue my study and evaluation of relevant prior art as these proceedings continue, at least up until the end of the period prescribed by the relevant rules and regulations. In the event further issues arise that concern the subject matter in my declaration or additional information becomes available, I expressly reserve the right to supplement and/or amend my opinions as stated in this declaration as well as the bases supporting my opinions. I specifically reserve the right to supplement and/or amend this declaration at such time that the Board defines the disputed terms in the Challenged Claims, particularly if the Board adopts alternate instructions not advanced by Petitioners. I further reserve the right to supplement and/or amend this declaration to submit a reply declaration based on any information or contentions expressed by Webroot or its experts attempting to rebut my opinions in this declaration. In addition, I reserve the right to use demonstratives or other exhibits at trial in support of my opinions in any of my expert declarations and/or supplements thereto.

X. CONCLUSION

193. For the reasons set forth above, it is my opinion that the Challenged Claims of the ’250 Patent are invalid.

Executed on March 8, 2023
Gainesville, Florida

By: */s/ Dr. Richard E. Newman*

Dr. Richard E. Newman
8524 SW 23rd Place
Gainesville, FL 32607
(352) 283-1083
nemo@ufl.edu