

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SOPHOS LTD. & SOPHOS INC.,

Petitioners

v.

OPEN TEXT INC.

Patent Owner

Case IPR2023-00699

Patent No. 8,418,250

**DECLARATION OF PROFESSOR NENAD MEDVIDOVIC, Ph.D.
IN SUPPORT OF PATENT OWNER'S RESPONSE**

TABLE OF CONTENTS

I.	QUALIFICATIONS	2
II.	UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES	7
III.	STATE OF THE PRIOR ART	9
IV.	The '250 Patent.....	11
	A. Summary Of The '250 Patent.....	11
	B. Claim Construction	19
	C. Level Of Ordinary Skill In The Art.....	22
V.	OVERVIEW OF PRIOR ART	24
	A. Overview Of Halperin	24
	B. Overview Of Mankins	28
	C. Overview of Kirby.....	30
VI.	NEITHER OF PETITIONER'S PROPOSED COMBINATIONS DISCLOSES OR RENDERS OBVIOUS THE CHALLENGED CLAIMS	31
	A. There Is No Motivation to Combine Halperin with Mankins	31
	1. Petitioner does not articulate any proposed combination of Halperin and Mankins	32
	2. A POSITA would not have been motivated to combine Halperin with Mankins	36
	3. A POSITA would not have identified Mankins as a possible solution to any shortcomings identified in Halperin.....	38
	4. Halperin discourages systems like Mankins's.....	42

B.	There Is No Motivation To Combine Halperin with Mankins and Kirby	45
C.	Independent Claims 1, 13, and 25.....	47
1.	Halperin Does Not Disclose Limitations [1P] to [1C], [13A] to [13C], [25P] to [25B], and [25I]	49
2.	The Petition’s Combination of Halperin in View of Mankins Does Not Render Obvious Limitations [1D] to [1I], [13D] to [13I], [25D], [25F], [25G], [25I] and [25J] Nor Dependent Limitations [2], [14] and [25B].....	60
VII.	CONCLUSION.....	70

EXHIBIT LIST

Exhibit	Description
2001	Declaration of Professor Nenad Medvidovic, Ph.D
2002	<i>Curriculum Vitae</i> of Professor Nenad Medvidovic, Ph.D.
2003	<i>Sonrai Memory Ltd. v. Kingston Tech. Co. and Kingston Tech. Corp.</i> , Case No. 6:21-cv-1284-ADA, Order Denying Kingston Tech. Co. and Kingston Tech. Corp.’s Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 94 (W.D. Tex., Oct. 18, 2022)
2004	<i>mCom IP, LLC v. Cisco Systems, Inc.</i> , Case No. 6:22-cv-00261-ADA, Order Denying Defendant’s Motion to Stay Pending <i>Inter Partes</i> Review, Dkt. 42 (W.D. Tex., Oct. 20, 2022)
2005	<i>Intellectual Ventures I LLC et al. v. Hewlett Packard Enterprise Co.</i> , No. 6:21-cv-00596-ADA, Order Denying Hewlett Packard Enterprise Company’s Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 104 (W.D. Tex., Dec. 22, 2022)
2006	Dani Kass, “Catching Up On Patent Litigation With Judge Albright” (Law360 Mar. 14, 2023); https://www.law360.com/pulse/articles/1582438/print?section=pulse/courts
2007	U.S. District Courts—Federal Court Management Statistics—Profiles—During the 12-Month Periods Ending March 31, 2018 Through 2023, “Table N/A—U.S. District Courts—Combined Civil and Criminal Federal Court Management Statistics (March 31, 2023)”, <i>available</i> https://www.uscourts.gov/sites/default/files/data_tables/fcms_na_distprofile0331.2023.pdf

Exhibit	Description
2008	<i>Webroot, Inc., Open Text Inc., v. AO Kaspersky Lab et al.</i> , Case No. 6:22-cv-00243-ADA-DTG, Order Granting Third Amended Scheduling Order, Dkt. 142 (W.D. Tex., Dec. 27, 2022)
2009	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Claim Construction Order, Dkt. 236 (W.D. Tex. Mar. 16, 2023)
2010	<i>Webroot, Inc. and Open Text Inc., v. Sophos Ltd.</i> , Case No. 6:22-cv-240-ADA-DTG, First Amended Complaint For Patent Infringement, Dkt. 42 (W.D. Tex. June 23, 2023)
2011	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Scheduling Order, Dkt. 39 (W.D. Tex. Aug. 17, 2022)
2012	<i>Webroot, Inc. and Open Text Inc., v. Trend Micro, Inc.</i> , Case No. 6:22-cv-00239-ADA-DTG, Defendant’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2013	<i>Webroot, Inc. and Open Text Inc., v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.</i> , Case No. 6:22-cv-00241-ADA-DTG, Defendant’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2014	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA-DTG, Defendant’s Preliminary Invalidity Contentions Appendix A (W.D. Tex., Sept. 14, 2022) (excerpted)
2015	Declaration of Professor Nenad Medvidovic, Ph. D. in Support of Patent Owner’s Response
2016	Transcript of deposition of Dr. Richard Newman taken December 7, 2023 (excerpted)
2017	Marius Gheorghescu et al., An Automated Virus Classification System, VIRUS BULLETIN CONFERENCE 2005, 294-300 (Oct. 2005)

I, Professor Nenad Medvidovic, Ph.D., declare as follows:

1. I have been engaged by King & Spalding LLP on behalf of OpenText Inc. (“Patent Owner”), to provide this Declaration concerning the technical subject matter relevant to Petitioner Sophos’s petition for Inter Partes Review (“Petition”) of U.S. Patent 8,418,250 (“the ’250 patent”).

2. I am over 18 years of age. I have personal knowledge of the facts stated in this Declaration and could testify competently to them if asked to do so.

3. I have reviewed and am familiar with the specification and the claims of the ’250 patent. In general, I will cite to the specification of a United States patent using the following formats: (Exhibit No. Col.:Line Number(s)) or (Exhibit No. Paragraph Number(s)). For example, the citation (EX-1001 1:1–10) points to the ’250 patent specification at column 1, lines 1–10. Also, for convenience, I use italics to denote limitations from the challenged claims.

4. All the opinions in this Declaration are based on the documents I reviewed and my knowledge and professional judgment. In forming the opinions expressed in this Declaration, I reviewed the Petition for *Inter Partes* Review of U.S. Patent No. 8,418,250 and supporting exhibits.

5. I have been asked to provide my technical opinions regarding how a person of ordinary skill in the art would have understood the claims of the ’250 patent at the time of the alleged invention. I have also been asked to provide my

technical opinions on how concepts in the '250 patent specification relate to claim limitations of the '250 patent. In reaching the opinions provided herein, I have considered the '250 patent, its prosecution history, and the references cited above and have drawn as appropriate on my own education, training, research, knowledge, and personal and professional experience.

I. QUALIFICATIONS

6. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will summarize briefly. A more detailed summary of my background, experience, and publications is set forth in my curriculum vitae (CV), which is submitted as Exhibit 2002.

7. I received a Bachelor of Science degree, *summa cum laude*, from Arizona State University's Computer Science and Engineering department.

8. I received a Master of Science degree from the University of California at Irvine's Information and Computer Science department.

9. I received a Doctor of Philosophy degree from the University of California at Irvine's Information and Computer Science department. My dissertation was entitled "Architecture-Based Specification-Time Software Evolution."

10. For nearly thirty years, my research has focused on the building blocks of software systems: software components (computational and data elements) and

connectors (interaction-facilitating elements), and mechanisms for their implementation. In particular, I have focused on middleware facilities and software frameworks. In my Ph.D. dissertation—whose first research results were published in 1995 while the dissertation was completed in 1999—yielded a new design style for the development and evolution of highly distributed software systems built out of independent components that exchange different types of information encoded in messages in order to accomplish specific goals. This research focused on the design principles behind those systems, the appropriate support for modeling the systems (both at the level of individual agents as well as their integrations into subsystems and entire systems), and the infrastructure for their implementation, deployment, and optimization. This work was applied across a wide range of domains, from traditional distributed systems to mobile devices and autonomous robots.

11. I am employed by the University of Southern California (“USC”) as a faculty member in the Computer Science Department and have been since January 1999. I am a tenured professor. Since July 2022, I have served as my Department’s Chair. Between January 2009 and January 2013, I served as the Director of USC’s Center for Systems and Software Engineering. Between July 2011 and July 2015, I served as my Department’s Associate Chair for Ph.D. Affairs. Between July 2021 and June 2022, I served as my Department’s Vice Chair for Tenure Affairs.

12. I teach graduate and undergraduate courses in software architecture, software engineering, and embedded systems. and I advise Ph.D. students. I have graduated nineteen Ph.D. students and advise seven students currently pursuing a Ph.D.

13. I served as Program Co-Chair for the flagship conference in my field—International Conference on Software Engineering (“ICSE”)—held in May 2011. I have served as Chair or Co-Chair for various other conferences in the Software Engineering field, including: the Fifth Working Institute of Electrical and Electronics Engineers (“IEEE”)/IFIP Conference on Software Architecture, the Third IEEE International Conference on Self-Adaptive and Self-Organizing Systems, the Fifteenth International Association for Computing Machinery’s (“ACM”) Special Interest Group on Software Engineering (“SIGSOFT”) Symposium on Component Based Software Engineering, the IEEE/CSSE/ISE Workshop on Software Architecture Challenges for the 21st Century, and the Doctoral Symposia at the Sixteenth ACM SIGSOFT International Symposium on the Foundations of Software Engineering and at the 44th International Conference on Software Engineering.

14. I serve or have served as an editor of several peer-reviewed journals, including “IEEE Transactions on Software Engineering,” “ACM Transactions on Software Engineering and Methodology,” “Journal of Software Engineering for

Robotics,” “Elsevier Information and Software Technology Journal,” “Journal of Systems and Software,” “Journal of Software Engineering Research and Development,” and “Springer Computing Journal.” Additionally, I have served as a guest editor of several special issues for other journals. Since January 2018, I have served as Editor-in-Chief of “IEEE Transactions on Software Engineering.”

15. Between September 2013 and September 2015, I served as Chair of the ICSE Steering Committee. Both prior to that (2008-2013) and subsequent to that (2015-present) I have been a member of the ICSE Steering committee. I am currently a member of the Steering Committee of the joint European Conference on Software Engineering and the Symposium on the Foundations of Software Engineering (“ESEC/FSE”). From July 2015 to September 2017, I chaired the Steering Committee of FSE, while it was a standalone symposium. I previously served as a member of the Steering Committees of the Working IEEE/IFIP Conference on Software Architecture.

16. Between July 2015 and June 2018, I served as Chair of the ACM SIGSOFT, the largest professional organization in my field. During the past year, I have chaired the committee in charge of ACM’s Software System Award, one of the most important technical awards given in my field.

17. I co-authored *Software Architecture: Foundations, Theory, and Practice*, a widely used textbook in the field of software systems architecture.

18. I have served as editor of various books in the Software Engineering field, including *Proceedings of the 3rd International Conference on Self-Adaptive and Self-Organizing Systems*, *Proceedings of the Warm-Up Workshop for the 32nd International Conference on Software Engineering*, *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture*, and *40 Editions of ICSE—the Ruby Anniversary Celebration*.”

19. I have authored or co-authored nearly 300 papers in the Software Engineering field. My most cited paper has been cited well over 3,000 times. A paper I co-authored in the 1998 International Conference on Software Engineering, my field’s flagship conference, was given ten years later, in 2008, that conference’s Most Influential Paper Award. Another of my publications received a Most Influential Paper Award in 2020. Several of my other publications have received “best paper” or “most cited” awards.”

20. Several of my publications have focused on different aspects of computer security. A paper describing a security-vulnerability detection tool for Android devices, built in my lab, won the Best Tool Paper award at the 2017 Automated Software Engineering Conference.

21. I have co-invented three patents in the area of cybersecurity.

22. I have been named a Distinguished Scientist of the ACM. I have been elected a Fellow of the IEEE. Fellow of the IEEE is IEEE's highest grade; less than 0.1% of its membership earns this grade each year.

II. UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES

23. I am not a lawyer, and I will not provide any legal opinions. Although I am not a lawyer, I have been advised that technical experts apply certain legal standards when they form opinions about the meaning and validity of patent claims.

24. I understand that a patent claim is invalid if it is anticipated or obvious in view of the prior art, and that a claim can be unpatentable even if all of the requirements of the claim cannot be found in a single prior-art reference. I further understand that invalidity of a claim requires that the claim be anticipated or obvious from the perspective of a person of ordinary skill in the art at the time the invention was made.

25. I have been informed that a patent claim is invalid if it would have been obvious to a person of ordinary skill in the art. In analyzing the obviousness of a claim, I understand the following factors may be taken into account: (1) the scope and content of the prior art; (2) the differences between the prior art and the claims; (3) the level of ordinary skill in the art; and (4) any so called "secondary considerations" of nonobviousness if they are present.

26. I have been informed that a claim can be obvious in light of a single prior art reference or multiple prior art references. I understand that to prove that prior art or a combination of prior art renders a patent obvious, it is necessary to:

- 1) Identify the particular references that, singly or in combination, make the patent obvious;
- 2) Specifically identify which elements of the patent claim appear in each of the asserted references; and
- 3) Explain why a person of ordinary skill in the art would have combined the references, and how they would have done so, to create the inventions claimed in the patent. I further understand that exemplary rationales that may support a conclusion of obviousness include:
 - Combining prior art elements according to known methods to yield predictable results;
 - Simple substitution of a known element for another to obtain predictable results;
 - Use of known technique(s) to improve similar devices (methods or products) in the same way;
 - Applying a known technique to a known device (method or product) ready for improvement to yield predictable results;

- Choosing from a finite number of identified, predictable solutions with a reasonable expectation of success, which I am informed is the “obvious to try” standard;
- Known work in one field of endeavor may prompt variations of the work for use in either the same field or a different field based on design incentives or other market forces if the variations are predicable to a person of ordinary skill in the art; and
- Some teaching, suggestion, or motivation in the prior art that would have led a person of ordinary skill in the art to modify the prior art reference or to combine the prior art reference teachings to arrive at the claimed invention.

27. I have been informed that, in considering obviousness, hindsight reasoning derived from the patent at issue may not be used.

III. STATE OF THE PRIOR ART

28. “Malware” generally refers to an executable computer file or an object that consists of or contains malicious code. Viruses, trojans, worms, and similar programs are all types of malware. (EX-1001 1:14-17.) Different types of malware work differently, and so what works to detect one type of malware (*e.g.*, a virus) may be less effective against another version (*e.g.*, a worm). (*See generally* EX-1001 1:14-2:56.)

29. At the time the invention in the '250 patent was invented, many systems designed to address the threat of malware focused on looking for previously known malware. For example, antivirus software might examine executable computer files for known signatures indicative of malware. (*E.g.*, EX-1001 1:18-29.) “Generally, the method of dealing with malware is that when new types of malware are released, for example via the Internet, these are eventually detected.” (EX-1001 1:22-24.) Providers of antivirus software push an update to their software intended to recognize and handle known malware. (EX-1001 1:24-28.) However, these types of systems suffered from inherent delay. (EX-1001 1:30-37.) A system would be vulnerable after release of a new piece of malware until such malware was detected and a solution generated and released to users. (*Id.*)

30. Other anti-malware systems sought to protect from previously unknown malware. For example, local area networks were known to implement a centralized computer responsible for authorizing the running of files on remote computers within the network. (EX-1001 1:60-67.) When a remote computer encountered a file that was unknown to it, it would forward a request to the central computer. (EX-1001 1:64-2:4.) But when the file was unknown even to the central computer, the central computer would obtain the whole file from the remote computer so it could determine whether it should be regarded as safe or malware. (EX-1001 2:4-7.) Such an analysis had to be “carried out manually or semi-

manually”—that is, not by the computer but by a human—by subjecting the file to a detailed analysis. (EX-1001 2:8-11.)

31. These analysis methods included, for example, emulation or interpretation, “which can still take days given the human involvement that is typically required.” (EX-1001 2:4-11.) Other prior art systems similarly relied on a human system administrator to “make a better informed though still ‘manual’ decision as to whether or not the file is safe to run.” (*E.g.*, EX-1001 2:18-27.) “There is therefore still a considerable period of time before a new file is classified as safe or as malware.” (EX-1001 2:12-13.)

32. Thus, prior art systems “rel[ie]d on deep analysis of a new object in order to determine whether or not the object is malicious, which introduces delay and therefore risk to users during the period that the file is analysed and new anti-malware signatures distributed[.]” (EX-1001 2:57-61.)

33. Accordingly, there was a long-felt need for an improved technique to efficiently determine whether objects were safe or malware.

IV. THE ’250 PATENT

A. Summary Of The ’250 Patent

34. The ’250 Patent is entitled “Methods and Apparatus for Dealing with Malware” and describes and claims a method and system for classifying a computer object as malware. (*E.g.*, EX-1001 Claims 1, 13, 25.) The ’250 patent describes a

computer “object” as including a computer file, partial file, sub-program, macro, web page, any piece of code to be operated by or on either the computer itself or an event within the computer. (EX-1001 7:63-66.) An “event” is an action or behavior, and according to the patent it includes actions that are “executed, emulated, simulated, or interpreted” in the computer. (Ex. 1001 7:65-66; 12:64-13:2)

35. Unlike the methods described in the prior art, the ’250 Patent describes a rapid determination of whether an object (according to the definition in ¶ 34) should be classified as safe or malware, without requiring a preliminary detailed analysis of the object. (EX-1001 3:8-18.)

36. The ’250 patent teaches a method that may be implemented in a distributed network, such as a local area network, having a “central” or “base” computer connected to multiple “remote” computers. (EX-1001 7:54-59, Fig. 1.)

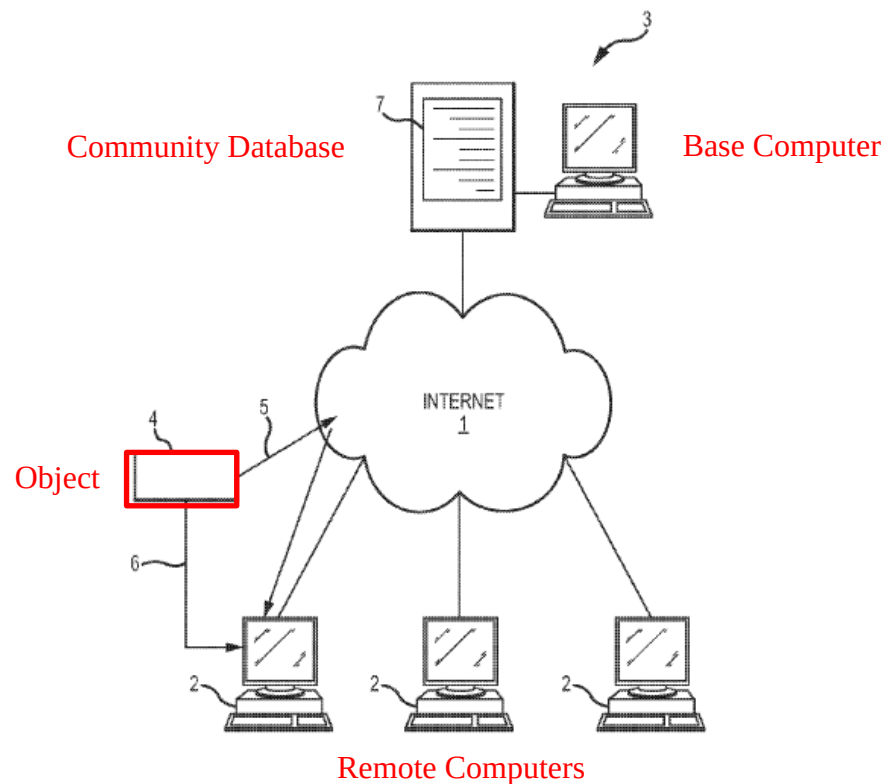


FIG.1

(EX-1001, Fig. 1, 7:54-8:16 (annotated).)

37. The '250 patent teaches that the remote computers provide data about an object to the base computer to facilitate a comparison of data at the base computer. (EX-1001 4:43-47.) Such data may include executable instructions within the object; the size of the object; the current name of the object; the physical and folder location of the object; 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. (EX-1001 4:52-61.)

38. One of the ways that this data may be provided to the base computer is in the form of a “key,” which the ’250 patent explains allows for a quick comparison of objects and faster analysis at the base computer. (EX-1001 3:51-4:6.) “The key preferably has at least one component that represents executable instructions contained within or constituted by the object.” (EX-1001 3:51-53.) This “important preferred feature” allows a comparison to be made at the base computer of only the executable instructions of the object. (EX-1001 3:53-55.) The ’250 patent explains that when objects with different names have the same executable instructions, this often indicates the objects are malware. (EX-1001 3:55-59.) The use of a key representing the object’s executable instructions means that the invention according to the ’250 patent treats these differently named but substantively similar objects as the same object for the purpose of determining whether they are safe or malware. (*Id.*)

39. Figure 2 is described as “a flowchart showing schematically the operation of an example of” the ’250 patent’s method. (EX-1001 7:46-48.)

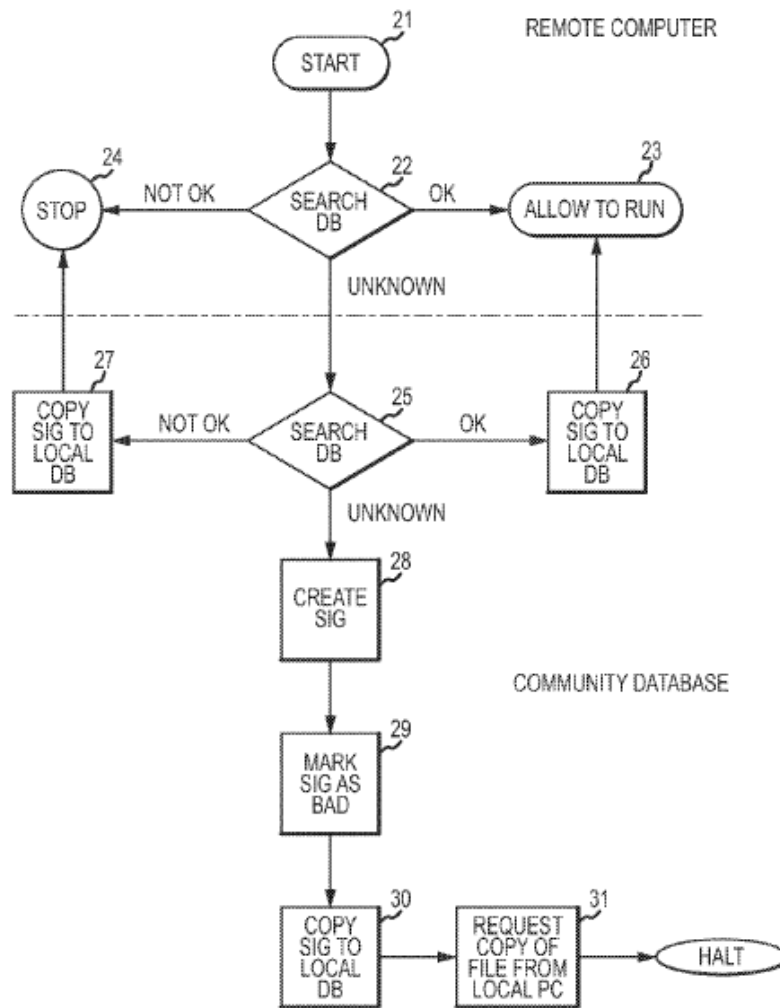


FIG.2

(EX-1001 Fig. 2, 8:16-15:5 (annotated). I describe the steps illustrated in Figure 2, below.

40. Starting at step 21, a computer object runs on a remote computer. (EX-1001 8:17-18.) At step 22, the remote computer searches its local database for a key representing that particular process, along with any related objects or events. (EX-1001 8:18-24, Fig. 2 (step 22).) If the process is known and a local key exists, the

remote computer allows or stops the process based on whether the key indicates that the process is safe or unsafe. (EX-1001 8:24-34, Fig. 2 (steps 23-24).)

41. When a search of the local database does not reveal the identity of the object, data about the object is passed along to the base computer. (EX-1001 8:45-47.) (This is illustrated between steps 22 and 25, as the process bridges the divide between the remote computer and the base computer. EX-1001 Fig. 2.) At step 25, the base computer searches the community database for the object's signature. (EX-1001 8:47-50, Fig. 2 (step 25).) If the base computer finds the object's signature in the community database, it may indicate to the remote computer that the object is safe (EX-1001 8:60-9:30, Fig. 2 (step 26)) or that it is unsafe (EX 1001 9:4-7, Fig. 2 (step 27)). This indication may determine whether the process is stopped or permitted to run. (EX-1001 Fig. 2, (steps 23-24, 26-27).) In either instance, the remote computer's local database may be updated with this additional information, so that the remote computer has this information immediately at hand the next time that computer encounters the object. (EX-1001 8:60-9:10.)

42. If the object remains unknown even after searching the community database, the '250 patent teaches creating a signature for the new object, and the object is initially marked as bad or unsafe. (EX-1001 9:11-17, Fig. 2 (steps 28-29).) Once the signature has been created, the '250 patent's invention provides it to the remote computer to copy to its local database, along with a message that might

instruct the remote computer not to run the object or might inform the user of the risk and offer the choice of running it; additionally, the community database may request a copy of the object itself, in case further analysis is needed. (EX-1001 9:17-25, Fig. 2 (steps 30-31).)

43. If the unknown object is allowed to run on the remote computer, the '250 patent teaches creating a "mask" to monitor the object's actions. (EX-1001 9:26-39, 14:55-61 ("The signature of an object may comprise or be associated with a mask.")) For example, the mask parameters might indicate "the expected behaviour of the object; any external requests or Internet connections that that object might legitimately have to make or call upon the remote computer 2 to make, including details of any ports or interfaces that might be required to be opened to allow such communication, any databases, either local or over a local area network or wide area network or Internet, that may be expected to be interrogated by that object; and so on." (EX-1001 14:61-15:4.)

44. In addition to monitoring processes that are not pre-authorized, the '250 Patent discloses an embodiment in which even pre-authorized objects are monitored while running to determine whether those objects exceed the parameters of the existing mask. (EX-1001 15:42-45, Fig 3 (steps 35 and 36).)

45. If the object's actions remain within the boundaries set by the mask, then the object is allowed to run. (EX-1001 15:45-47, Fig. 3 (step 34).) But if the

object's actions extend outside the mask—if, for example, the object attempts to initiate connections that have not been pre-authorized—then the behavior is flagged at an alert step and the '250 patent teaches that subsequent action may be taken. (EX-1001 15:47-51, Fig. 3 (step 37).)

46. The object undergoes continuous monitoring, which allows the mask to be updated when current or future object behavior varies from prior object behavior. (EX-1001 15:64-16:3.) For example, when an object exceeds the mask's initial parameters but no ill effects follow, or if the system recognizes the behavior from other objects and knows that it is safe, then the '250 Patent teaches modifying the mask to include the new behavior as a safe or allowed behavior. (EX-1001 15:9-21.) Continuous monitoring allows for similar reclassification of the object as unsafe if the object violates the parameters of the generated mask by attempting to connect to a known unsafe website or take an action known to be suspicious (*i.e.*, an action only attempted by malware). (EX-1001 15:22-25.)

47. Thus, the '250 Patent describes and claims systems and methods that provide more efficient and accurate determinations of whether an object is safe or malware. The '250 patent's "sophisticated pattern analysis" allows for "rapid determination of the nature of the object to be made, without requiring detailed analysis of the object itself." (EX-1001 3:8-18.) Its methods and system decreased reliance on signature verification and negated the need for manual, time-consuming

“deep analysis” of an object. (*Id.*) This improved system performance and reduced the number of resources required, allowing for a new method of computer object classification that is rapid, efficient, and accurate.

B. Claim Construction

48. Petitioner does not construe any claim terms in its Petition. However, Petitioner and Petitioner’s expert rely on interpretations of the ’250 patent that venture beyond the plain and ordinary meaning of the claim terms. Accordingly, I provide below my interpretation of these terms and an explanation of why Petitioner’s interpretation in the context of the ’250 patent is flawed.

49. Petitioner seems to interpret the “classifying” term (as in claim 1) as not precluding classification performed by a human. I disagree, because classification performed by a human is not consistent with the plain and ordinary meaning of the term and contradicts the teachings of the patent.

50. A POSITA would have understood “classifying” to preclude classification by a human, because around 2006 (the priority date of the ’250 patent), manual classification was commonplace. (EX-2017, 294). Contemporaneous sources to the ’250 patent explain that manual classification is well-known and acceptable. *Id.* Research initiatives and inventions at the time were focused on new techniques in automated classification and naming. *Id.* As a result, it would not have been logical to claim a “method of classifying a computer object as malware” (as in

the '250 patent) based on a manual classification. The only reasonable interpretation for a POSITA reading a patent claiming a novel invention is that said classification was performed automatically and not according to the ubiquitous and well-known manual classification techniques.

51. Petitioner's interpretation that "classifying" can include classification performed by a human is also contradicted by the patent. The '250 patent's specification highlights the known shortcomings of manual classification in the prior art. For example, manual classification incurs delays to account for observation and analysis by a human. This issue was not far-fetched. Indeed, it was well known. (EX-2017, 294 (explaining that "the increasing number of viruses and virus variants and the need to reduce time-to-market necessitate anti-virus (AV) vendors to reduce the time spent in investigating the origins of samples.")) A POSITA reading the specification of the '250 would understand that the '250's criticism of manual techniques implies that the "classifying" claimed in the limitations of the patent would not be performed manually. These very shortcomings are the problem that the system claimed in the '250 patent was intended to solve.

52. I also understand that Petitioner's expert relies on an interpretation of the term "computer object" as including computer "events." This far-fetched interpretation runs contrary to what a POSITA would understand and contrary to the definitions provided in the patent. Moreover, upon reading Dr. Newman's

declaration, I was not able to identify a single instance in which Dr. Newman explained or otherwise justified his understanding that a “computer object” in the context of the ’250 patent included a computer event.

53. The ’250 patent makes clear that a computer object is defined as one of five things: (1) a computer file; (2) a part of a file or a sub-program; (3) a macro; (4) a web page; or (5) any other piece of code. (EX-1001 7:63-66.) This is consistent with my understanding of what a computer object is and what a POSITA would have understood a computer object to be. In this sense, an object is a file or program that can be operated by or on the computer. Because, as I explained above, the ’250 patent is concerned with classification of computer objects, computer objects must be easily categorized, recorded, and reclassified (if necessary). Petitioner’s interpretation that a computer object can be an event is not reconcilable with the definition provided in the patent.

54. The ’250 patent also provides a definition for the term “event.” (EX-1001 12:64-13:2.) Consistent with my understanding of an event and what a POSITA would have understood, an “event” is an “action[] or behaviour[] of an object acting upon another object or some other entity.” *Id.* That is, objects are operated by or in response to, or operate on events. As the ’250 patent explains, an event “has three principal components: the key of the object performing the act (the

‘Actor’), the act being performed (the ‘Event Type’), and the key of the object or identity of another entity upon which the act is being performed (the ‘Victim’).” *Id.*

55. As Dr. Newman also explains, a POSITA such as himself would understand a “computer object” to “refer to ... things only like files or pieces of files.” (EX-2016 49:115-17, 112:18-22.) I concur. A computer object, in the context of the ’250 patent refers only to the five categories of files, programs, and codes listed explicitly in the patent. Any attempt to broaden the definition of computer object to encompass Halperin’s interaction with computer events is unsupported and should be rejected.

56. I understand that in the District Court’s March 16, 2023 claim construction order, the Court determined that no terms of the ’250 patent were indefinite and that all terms should be given their plain and ordinary meaning. (EX-2012 3-5, 8.) Consistent with that order, for all other terms except for those explicitly mentioned above, I apply the plain and ordinary meaning of all claim terms.

C. Level Of Ordinary Skill In The Art

57. I understand that certain issues relating to the validity of the ’250 patent must be judged from the perspective of a person of ordinary skill in the relevant art, as I discuss below. I have been asked to define the level of a “person having ordinary skill in the art” or “POSITA” that someone would have had at the time the alleged invention as claimed was made. In deciding the level of ordinary skill, I have

considered the following factors, which I have been informed are relevant in determining the level of ordinary skill in the art:

- The levels of education and experience of persons working in the field;
- The types of problems encountered in the art;
- The prior art solutions to these problems;
- The rapidity with which innovations are made; and
- The sophistication of the technology.

58. A POSITA at the time of the invention of the '250 patent (which claims priority to an earlier application (GB 0513375.6) filed in Great Britain on June 30, 2005) would have had a Bachelor's degree in an accredited program of electrical engineering, computer engineering, computer science, or a similar discipline, and 2–3 years of practical work or research experience in the general fields of electrical engineering, computer science, networking, communications, and device and network security. More advanced degrees or training in a related discipline or both can compensate for shorter work experience.

59. I understand that Petitioner has defined the level of ordinary skill in the art as:

As of June 30, 2005, a person having ordinary skill in the art (POSITA) would be a person having a bachelor's degree in computer science, computer engineering, or an equivalent, as well as two years of industry experience and would have had a working knowledge of host

monitoring systems, software security analysis, and dynamic malware analysis. Additional graduate education could substitute for professional experience, or significant experience in the field could substitute for formal education.

(Petition, 9.)

60. My opinions as expressed in this Declaration would be the same when viewed under either party's definition of the level of ordinary skill in the art. The minor variations between them are immaterial.

61. Based on either standard, I was at least a person of ordinary skill in the art at the time of the invention of the '250 patent.

V. OVERVIEW OF PRIOR ART

A. Overview Of Halperin

62. Halperin is directed to detecting patterns of collective computer behavior on a network that suggest the presence of viral activity and taking containment actions to halt further spread. (EX-1006 Abstract, [0036-0057].).

63. Halperin observed that some prior art antivirus software identified virus behavior by characterizing a program's legitimate behavior and identifying any deviations from such behavior. (EX-1006 [0006].) Halperin criticized this type of object-specific technique as often resulting in false alarms due to the fact that computer user behavior tends to vary over time and between users. (EX-1006 [0006].) Halperin found that this technique's false-positive shortcomings led users

to either disable this type of antivirus software entirely or set its sensitivity so low that it failed to find existing viruses. (*Id.*)

64. Halperin further recognized that “[i]n most cases, viruses are spread when computer users send infected files to other computer users,” for example, via e-mail, data storage media (*e.g.*, disk or CD), or by sharing infected files from one computer to another. (EX-1006 [0003].) Some viruses were even capable of spreading on their own “with little or no intervention on the part of the computer user.” (EX-1006 [0004].) For example:

A virus that spreads via e-mail messages will typically access an e-mail program’s address book or sent/received mail folders and automatically send itself to one or more of these addresses. Alternatively, the virus may attach itself to otherwise innocuous e-mail messages that are sent by a computer user to unsuspecting recipients.

(EX-1006 [0004].)

65. Thus, Halperin sought to provide a network-level method to detect symptoms of malware and take containment actions to prevent the possibility of malware spreading, while overcoming the “false alarm” problem it perceived in the prior art. (EX-1006 [0037].) Rather than monitor or analyze a particular object—like the ’250 patent—Halperin’s proposed solution detects suspicious patterns of “target behavior” across a group of computers on a network (indicating the presence of a potential virus in the network) and takes one or more of what Halperin calls “virus

containment” actions. (EX-1006 Abstract, [0038]-[0057].) Figure 6 illustrates this overall process:

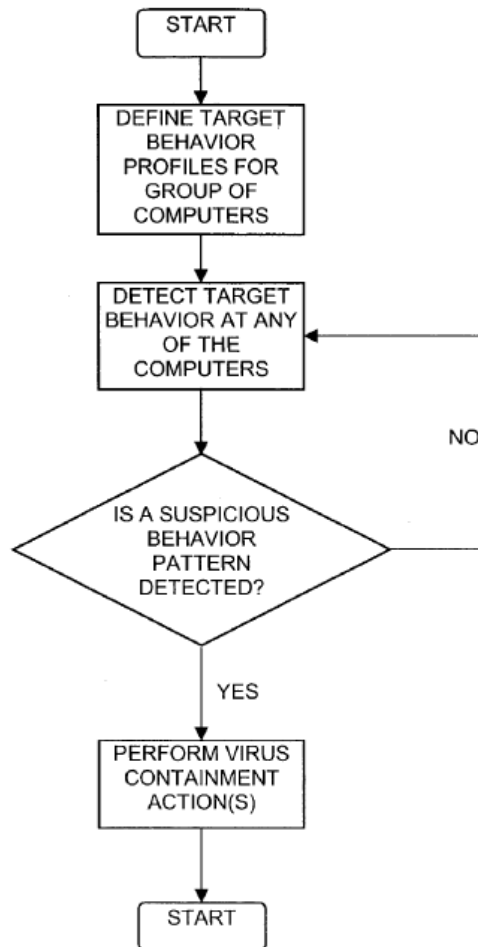


Fig. 6

(EX-1006, Fig. 6.)

66. Halperin starts by defining one or more “target behavior profiles” for a group of computers on a network. (EX-1006 [0038], Fig. 6.) Examples of target behavior profiles include, among others, “[s]ending messages to more than a predefined number of users during a predefined period of time;...[m]odifying

operating system file such as the Microsoft Windows registry; [and] [d]eleting more than a predefined number of files on the computer's hard disk during a predefined period of time.” (EX-1006 [0039]-[0046].) The '250 patent examines the *same individual object*—for example, an individual computer program—across multiple computers on a network. By contrast, Halperin seeks the *same computer behavior* being exhibited by multiple computers on a network. (EX-1006 [0037]-[0038].)

67. Halperin then employs a server on the network to correlate the presence of target computer behavior on multiple computers, *e.g.*, by “identifying the same *activity* in different computers at about the same time.” (EX-1006 [0047].) Halperin then teaches determining whether this collective computer behavior corresponds to a “predefined suspicious behavior pattern,” which would indicate that a virus may have infected the network. (EX-1006 [0047].)

68. The exemplary patterns of suspicious behavior Halperin discloses all relate to a “certain percentage of the computers in the network” all doing the same thing, most often within a relatively short amount of time. (EX-1006 [0048]-[0056].) For example, Halperin states that when a “certain percentage of computers in the network send[] more than 10 messages per minute in the last 5 minutes,” a virus may be present on the network. (EX-1006 [0048].) That is, whereas the '250 patent relies on data that is specific to a particular computer object to determine whether that object is malware (Section IV(A), *supra*), Halperin relies on data that is not

particular to any object, because it seeks to determine whether a certain percentage of computers on the network are behaving unexpectedly.

69. Halperin teaches that if the server detects a suspicious behavior pattern, it may initiate “virus containment actions.” (EX-1006 [0057], [0023]-[0030].) The system proffers no means of identifying a particular object as the malware culprit, much less classifying any object as malware or not malware. Instead, Halperin’s proposed solutions amount to either telling a human that the system has discovered symptoms indicating that malware might be present (*id.* [0025]–[0027]) or halting the flow of information across the network (*id.* at [0024], [0028]–[0030].) Halperin does not teach removing the offending software—and could not, because it does not teach how to identify the offending software. (*See* EX-1001 [0024]-[0030].)

70. Thus, whereas the ’250 patent examines information about *the same object* across multiple computers on a network to determine whether the *object* should be classified as malware, Halperin looks for *the same behavior or activity* across multiple computers on a network to determine whether the *network* shows signs of a malware infection, causing the Halperin system to take generalized containment actions. (EX-1006 [0037]-[0038].)

B. Overview Of Mankins

71. Mankins is directed to a kernel-based intrusion detection system using Bloom filters. (EX-1008 Abstract.) The Bloom filter is a data object, embedded in

the operating system, that contains data representative of expected system-call behavior associated with a computer program. (EX-1008 2:41-52.)

72. A Bloom filter is generally known as a mathematical construct directed at determining whether a value is within a specified set of values. The Bloom filter is generated by running a specified hash function on each value that is in the set. Using the same hash function on the value being assessed for set membership yields a result that can be quickly compared to the Bloom filter value to return a result. The result indicates either that the value is *definitely not* within the set or that the value *may be* within the set. That is, Bloom filter results do not include false negatives (in this context, mistakenly identifying non-malicious system calls as malicious), but may include some number of false positives (in this context, mistakenly identifying malicious system calls as non-malicious). The rate of false positives is calculable based on the size of the Bloom filter, and that rate may be reduced by increasing the size of the Bloom filter.

73. In a training phase, Mankins creates a Bloom filter object by monitoring normal operation of the computer program when it is not subject to intrusion attacks. (EX-1008 3:37-40.) The Bloom filter tracks individual system calls or system-call sequences, such as “open, read, mmap, mmap, open.” (EX-1008 3:31-33, 6:20-23.) By compacting the data representing normal system-call behavior of an object using hash functions to construct a Bloom filter representing the collection of typical

system calls, Mankins is able to quickly determine if a program operating in “real (i.e., actual) circumstances” is executing anomalous system calls that may suggest “intrusion attempts.” (EX-1008 3:46-67.)

74. Notably, Mankins makes no determination of what any given sequence of system-calls achieves. That is, Mankins focuses entirely on the system calls themselves and is completely agnostic to the behavior of the computer as a result of the system calls.

C. Overview of Kirby

75. Kirby is directed to “integrated” malware protection, combining a behavioral approach with a signature-matching approach. (Ex. 1007 Abstract).

76. Kirby’s behavior-monitoring approach is designed to determine whether actions a piece of software takes are indicative that the software is malware. Under that approach, Kirby determines whether actions a piece of software takes match a predefined “behavioral heuristic of actions known to be harmful or undesirable.” (Ex. 1007 7:48-50, 10:54-60.)

77. Kirby teaches a separate, parallel signature-matching approach, which it describes as “conventional virus detection.” (Ex. 1007 at 7:30-40.) In that approach, Kirby teaches comparing the contents of a piece of software against a predefined “cache” of known malware objects. (Ex. 1007 at 9:31-40.)

VI. NEITHER OF PETITIONER'S PROPOSED COMBINATIONS DISCLOSES OR RENDERS OBVIOUS THE CHALLENGED CLAIMS

78. The Petitioner and its expert contend that the combination of Halperin and Mankins renders obvious every limitation of claims 1, 12-14, and 24, and some limitations in claim 25. The Petitioner and its expert additionally allege that Halperin in view of Kirby and Mankins renders obvious every limitation of claims 1-30.

Ground 1	Section 103(a)	Obvious over Halperin in view of Mankins	Claims 1-2, 12-14, 24
Ground 2	Section 103(a)	Obvious over Halperin in view of Kirby and Mankins	Claims 1-30

79. I disagree with the arguments in the Petition and the opinions set forth by Dr. Newman for at least the reasons that I discuss in further detail below.

A. There Is No Motivation to Combine Halperin with Mankins

80. It is my opinion that a POSITA would have had no motivation to combine Halperin with Mankins, and I disagree with Dr. Newman's opinion to the contrary. First, Petitioner does not adequately explain its proposed combination, instead only identifying individual concepts disclosed in each reference. As a result, I am forced to base my analysis not on a combination but on a loose collection of disparate concepts from each of Halperin and Mankins. Second, I disagree with the reasons that Petitioner and Dr. Newman provide regarding *why* a POSITA would

have been motivated to bring together these two references—and, to the contrary, in my view a POSITA would not have been motivated to combine them. Third, I disagree with Dr. Newman that a POSITA seeking to combine Halperin and Mankins would have had a reasonable expectation of succeeding in realizing such a combination.

1. Petitioner does not articulate any proposed combination of Halperin and Mankins

81. Petitioner has not presented a coherent explanation of its proposed combination of Halperin in Mankins. I cannot discern from the petition what parts of Halperin and what parts of Mankins Petitioner proposes combining, nor can I identify what Petitioner’s proposal is for the combined system.

82. At each turn, both in Petitioner’s general “Motivation to Combine and Reasonable Expectation of Success” section and in each of the “Analysis” sections for the limitations, Petitioner merely contends that either Halperin or Mankins (or both) disclose a particular feature. Because of this insufficient disclosure, I am forced to guess how a combined Halperin-Mankins system would function according to the Petition.

83. At the outset, Petitioner suggests two general approaches to integrating the two systems—neither of which aligns with how Petitioner describes each of the references in the limitation-specific analysis. One approach incorporates a Bloom filter at the base computer; the second incorporates a Bloom filter at a remote

computer. (Petition 22–23.) Yet in each of the specific limitations of the independent claims in Ground 1, Petitioner relies on Mankins alone only for the mask and monitoring limitations (1E, 1F, 1G, and corresponding limitations in claim 13). Whether Mankins is implemented centrally or in a distributed fashion has no bearing on at what point in the Halperin process Mankins runs or how it interacts with the elements of the Halperin system. Thus, neither of these general approaches provides a legitimate (even if general) explanation of the proposed combination.

84. Dr. Newman attempts to clarify the Petition’s combination by offering more detail of the proposed implementation of a system combining Mankins with Halperin. Dr. Newman claims that “the Mankins Bloom filter process would begin where the Halperin ‘predefined suspicious behavior pattern’ check did not identify malware.” (EX-1004, ¶68.) At the outset, as a proposed combination, this sentence is extremely vague. First, Dr. Newman does not explain how a POSITA would determine when the “Halperin ‘predefined suspicious behavior pattern’ check did not identify malware.” *Id.* The Halperin system is designed to run and observe network behavior continuously and in real time. The Halperin system never makes a final determination or classification that malware has not been detected. There is no point at which Halperin stops scanning for suspicious behavior, because suspicious behavior could appear on the network at any time. This is unlike the ’250 patent which may classify a specific computer object and then end its investigation

of that object (as in claim limitation 1C). I discuss this in further detail below with regard to claim limitations [1C] and [1I]. In short, under Dr. Newman's proposal, Mankins would never operate because Halperin never performs a discrete "predefined suspicious behavior pattern" check." Second, Dr. Newman does not explain whether Halperin's system would continue to run or would cease operation once the Mankins process begins. Third, assuming that Halperin's process did halt and cede control to Mankins, Dr. Newman never explains how the Halperin process would continue to observe network behavior at some point in the future.

85. Dr. Newman's proposal is not a combination, but rather an application of the "defense-in-depth strategy," since Halperin and Mankins would operate entirely independently of each other.¹ Defense-in-depth refers to the cybersecurity strategy of layering multiple different defense strategies, often in succession, to achieve greater protection; however, defense-in-depth requires careful coordination between the layers of defense, such that the layers operate with synergy to increase defense. When care is not taken and multiple security strategies are ineffectively combined, the result is not added security but instead redundancy, waste of resources, and frequently an increase in false detections or decreased protection. For

¹ Dr. Newman even explicitly recognizes this, because he characterizes the so-called "combination" as "performing [an] extra test" when referring to the execution of Mankins. (EX-1004, ¶69.)

example, unless two applications employed in a defense-in-depth strategy are carefully coordinated, each may waste resources offering redundant protection against the same malware strategy while failing to address a second malware strategy entirely. The user of the system would thus obtain no benefit from, but would suffer the significant resource cost of, running two applications. Here, a POSITA would not have considered running Halperin and then Mankins as two separate malware detection layers to be a useful defense-in-depth combination because the two layers would be unable to share or benefit from each other's data and might even contradict each other. Mankins would be an arbitrary addition to Halperin, providing no discernable benefit to its operation.

86. Nor do I understand—and, again, Dr. Newman does not explain—how a POSITA would determine *which* object or objects should be analyzed according to the system in Mankins. In a situation where a Halperin system failed to detect *any* symptoms of malware infection, a POSITA would not even have an indication of which computers to focus on, much less any way to determine which objects on those computers should be run first to train a Mankins Bloom filter and then to monitor those objects' system calls. Because of the large number of objects running on an indeterminate number of computers distributed on a network, I disagree with Dr. Newman's contention that this would result in a “relatively small” “additional cost.” (EX-1004, ¶ 69.) To the contrary, this ill-fitting combination would make for

an extremely resource-intensive approach to detecting malware, because a Mankins add-on would need to generate and run a Bloom filter for *every* piece of software running on *every* remote computer, and then report the results back to the Halperin central computer. Again, because Halperin provides no context clues as to which programs may be responsible for any suspicious target behavior, there is no way to make that determination without examining every single program on every remote computer that exhibited suspicious behavior. This is exacerbated even further when Halperin fails to detect *any* symptoms of malware infection.

2. A POSITA would not have been motivated to combine Halperin with Mankins

87. Integrating Mankins wholesale into a Halperin system would be extraordinarily resource intensive. A POSITA seeking to improve Halperin would not have sought the approach described in Mankins.

88. Besides the shortcomings in the implementations that Dr. Newman suggests, a POSITA would not have been motivated to combine Halperin and Mankins because a POSITA would recognize that the combination could lead to decreased accuracy overall. Dr. Newman argues that as long as a combination “does not raise excessive false positives, it is a useful tool.” (Ex 1004, ¶ 70.) But, Dr. Newman disregards the fact that false positives are not the only potential errors in a malware detection system. The combination of Mankins with Halperin would not improve Halperin because it would fail to detect errors in the operation of Halperin

and could introduce increased inaccuracies from which Halperin would not suffer if uncombined. Thus, a POSITA would not have been motivated to combine Halperin with Mankins in any way, regardless of how the combination was actually implemented.

89. Mankins would not reduce false positives in Halperin. At the outset, Halperin warns that when “characterizing legitimate software behavior and ... identifying any deviation from such behavior[,] false positives arise” because “computer user behavior is quite dynamic and tends to vary over time and between different users.” (EX-1006 [0006].) To rectify this issue, Halperin employs a central server to detect a threshold number of computers performing the same suspicious behavior before it signals that suspicious activity is detected. In this way, the Halperin threshold can be increased to decrease false positives (and detection sensitivity as a result), or vice versa. Because Mankins operates at an object level, Mankins can reduce false positives in a Mankins-*only* system by monitoring Bloom filter data objects together (EX-1008 10:53-64.) But the use of Mankins would not affect the threshold mechanism of Halperin, which would remain at whatever predefined level to which it was originally set. In this way, the risk of false positives in Halperin remains unchanged by the use of Mankins.

90. Further, a POSITA would recognize that combining Halperin with Mankins would be problematic even when Halperin is *not* making faulty detections.

For example, if Halperin accurately detects that there is no viral outbreak on the network, but a Mankins Bloom filter data object (mistakenly) detects a suspicious behavior on an individual remote computer, the system would be forced to resolve this discrepancy between the two sources of data. This is a fundamental incoherency between the object-focused approach of Mankins and the network-level behavioral approach of Halperin. Dr. Newman offers no way of discerning which system's results should prevail and, in fact, a POSITA would have their confidence minimized in the proposed combination.

91. As a result, neither of the technical implementations that Dr. Newman offers is motivated. Both implementations run contrary to the teachings of the references and also suffer from technological shortcomings that a POSITA would recognize would be prohibitive.

3. A POSITA would not have identified Mankins as a possible solution to any shortcomings identified in Halperin

92. As I described above, a POSITA seeking to improve Halperin would not have sought the approach described in Mankins because it would have been technically ineffective in providing any advantages to Halperin. In addition, none of Petitioner's proposed reasons to allegedly improve Halperin are substantiated by Halperin's operations. I do not discern any reasons, in either the petition or Dr. Newman's declaration, for a POSITA to seek to improve Halperin.

93. Petitioner offers only the suggestion that “a POSITA would have recognized that using Mankins’s mask approach in a distributed system such as Halperin’s would improve the system by reducing false positives.” (Petition 34.) But Mankins cannot reduce false positives in Halperin in the system Petitioner proposes. A false positive in Halperin is a warning that suspicious behavior has been detected on the network, when in fact no malware is present. (See EX-1006 ¶ 75.) Meanwhile, as discussed above, Petitioner and Dr. Newman explain that “the Mankins Bloom filter process would begin where the Halperin ‘predefined suspicious behavior pattern’ check did not identify malware.” (Ex. 1004, ¶68; see also Petition 23 (“if the system does not identify malware based on the behavioral data received from remote ‘computers 500,’ Mankins’s process would begin”).) In this circumstance, there would be no (false) “positive” detection in Halperin for Mankins to correct. Namely, if Mankins only runs when Halperin *does not identify* malware, Mankins has no opportunity to correct a misidentification of malware. Thus, Dr. Newman’s suggestion—that seeking to reduce false positives would motivate a POSITA to combine the two references as described in the Petition—is incorrect.

94. Further, Petitioner contends—without any support from Dr. Newman—that “incorporating Mankins’s Bloom filter data object into Halpern’s malicious activity detection approach would have been nothing more than a

predictable use of a prior art element according to its established function” because a POSITA “would have recognized that the examples of ‘target behavior’ described in Halperin ... would be detected by monitoring system calls.” (Petition 21.) Not so.

95. Halperin’s sensors aggregate behavior across the network to look for suspicious patterns. (EX-1006 ¶¶ 48-56.) Each individual behavior alone is not suspicious. For example, if any one computer deletes a file by a particular name, this is not suspicious. However, if “[a] certain percentage of computers in the network delet[e] a file by the same name within the last 1 hour,” Halperin considers this suspicious. (EX-1006 ¶¶ 51.) Mankins, on the other hand, does not describe any aggregation of the results of different Bloom filters. Instead, Mankins pinpoints individual actions indicative of malware by detecting anomalous system calls. When the Bloom filter data object for a particular behavior of a particular program does not contain a match, Mankins notifies the user of an anomalous behavior. (EX-1008 12:7-14.) Because Mankins is narrowly focused on anomalous system calls of individual programs, it is not an effective fit with Halperin, which *ignores* individual anomalies or actions unless they occur in the aggregate across the network. Petitioner does not explain how Mankins’s Bloom filter data objects would have their anomaly detection aggregated or, alternatively, how Halperin would utilize single-anomaly detection mechanisms. A POSITA would therefore not find it a

“predictable use” of Mankins to employ its single-anomaly detection mechanism in an aggregated-anomaly system.

96. Petitioner also suggests that it would be “an obvious modification to collect system call sequences as part of the data” Halperin receives. (Petition 22.) But this argument contradicts how Petitioner contends the supposed combination of Mankins and Halperin (or their further combination with Kirby) discloses the limitations of the ’250 patent. In both of its proposed combinations, Petitioner contends that Halperin alone teaches “receiving data”; it does not rely on Mankins in combination with Halperin for that limitation. (Petition 25.)

97. Lastly, Mankins is an unintuitive place for a POSITA seeking to improve Halperin’s system to turn to. Mankins’s Bloom filter data objects are specific to an object and uniquely created at each remote computer. (EX-1008 10:57-64.) As a result, sometimes a behavior matches a Bloom filter (*i.e.*, the computer thinks the behavior is benign) when the action should have not matched the Bloom filter and triggered an alert.² (EX-1008 8:61-64.) Because of this false-negative issue with Mankins, if combined with Halperin, Halperin risks detecting fewer behaviors at end-point computers than it should, causing Halperin to

² Although Mankins defines this as a false positive (because it is looking for system-call *matches*), in the context of Halperin (in which there is no filter to match against) this would, in fact, be a false negative.

incorrectly believe there is no malware outbreak. Accordingly, Mankins would not provide a benefit to Halperin, whether to detect false positives or false negatives in Halperin. So, a POSITA seeking to improve Halperin's ability to detect anomalous behavior would not augment Halperin with a system *also* susceptible to failing to detect malicious activity.

4. Halperin discourages systems like Mankins's.

98. I read Halperin to expressly discourage detecting malware by employing heuristic techniques that are based on an object's normal behavior. In the background, Halperin criticizes some previously known malware detections techniques. One example Halperin criticizes is signature matching. Halperin explains that signature matching suffers in that signature matching is not effective against new viruses. (Ex. 1006, ¶ 5). A second method Halperin criticizes is what it calls "software that employs heuristic techniques." Halperin defines such disfavored techniques as those that "identify typical virus behavior by characterizing legitimate software behavior and then identifying any deviation from such behavior." (Ex. 1006, ¶ 6). Halperin cautions that because "computer user behavior is quite dynamic and tends to vary over time and between different users[,] [t]he application of heuristic techniques ... often results in a false alarm whenever a user does anything unusual." (Ex. 1006, ¶ 6.) Halperin explains that as a result of this, "computer users ... disable such software or set the sensitivity of such software so low to the

point where new viruses are often not identified.” (Ex. 1006, ¶ 6.) Accordingly, it is my opinion that a POSITA reading Halperin would have been discouraged from relying on either signature matching or normal-behavior-based heuristics for viral detections.

99. I read Mankins to teach exactly what Halperin discourages—namely, a system that “identif[ies] typical virus behavior by characterizing legitimate software behavior and then identifying any deviation from such behavior.” (Ex. 1006 ¶ 6.) Mankins teaches implementing Bloom filters in a computer’s kernel. Mankins’s Bloom filters, like those I am generally familiar with in the art, essentially map out an object’s expected system-call sequence—what Halperin refers to as “legitimate software behavior.” Using Bloom filters, Mankins’s monitoring subsystem, which I understand runs at the kernel, “monitor[s] system-call sequences and compare the monitored sequences with predefined normal system-call sequences to determine whether an anomalous sequence of system calls has occurred in the OS kernel.” (Ex.-1008 6:30-34.) In accordance with my understanding of Bloom filter data objects, Mankins explains that the monitoring happens “during operation of a particular computer program.” (Ex.-1008 6:35-39.) Mankins’s process of using its Bloom filter to determine whether a sequence of system calls is “anomalous” is what Halperin describes as “identifying any deviation from [legitimate software] behavior.” It is

my opinion, therefore, that Mankins teaches precisely the kind of “heuristic technique[]” that Halperin considers problematic.

100. Just like Halperin cautions, it is my opinion that Mankins produces false positives. I even read Mankins to acknowledge this fact. For example, Mankins recognizes that the complexity of the Bloom filter’s hash function and the size of the bit vector (*i.e.*, the amount of data each Bloom filter can store) determine the likelihood of detecting a false positive or false negative. (Ex.-1008 8:38-42.) Said another way, I understand that at some point Mankins’s Bloom filters will trigger a false positive or a false negative. Incorporating Mankins’s Bloom filter into Halperin’s system would not improve this reality. What is more, incorporating Mankins’s Bloom filter into Halperin’s system would actually increase the number of false negatives that either system would have encountered individually. For example, because Halperin triggers an alarm only upon a threshold number of computers performing the same suspicious behavior and because Mankins’s system may miss an otherwise abnormal behavior, it is even more likely that Halperin’s system will not trigger an alarm when it should.

101. Based on my understanding of Halperin and Mankins, it is my opinion that a POSITA would have been discouraged from combining the two references by any means. Halperin discourages employing normal-behavior-based heuristics and, instead, relies on detecting a pattern of suspicious behaviors by a threshold number

of computers. It is, therefore, my opinion that a POSITA would not have considered modifying Halperin to focus on normal behaviors of an object—at any stage of Halperin’s detection process.

B. There Is No Motivation To Combine Halperin with Mankins and Kirby

102. Because there is no motivation to combine Halperin with Mankins, Petitioner’s second proposed combination—which combines Halperin with not just Mankins but also Kirby—is also unmotivated.

103. Petitioner’s description of its Halperin-Mankins-Kirby combination is even less well developed than its Halperin-Mankins description. (See Petition 46-47.) On the basis of that perfunctory description, I am unable to ascertain how Petitioner expects that any system combining the teachings of these three references would in fact operate. Petitioner suggests that a combination would “incorporat[e] Kirby’s signature matching techniques to aid in the classification of objects,” but I am unable to identify how Petitioner proposes to fit those techniques into either Halperin alone or into a Halperin-Mankins combination. In my view, a POSITA would have expected that incorporation to take place during the time in which the system in question is identifying a piece of software as malware. But Petitioner does not rely on Kirby for the claim limitations in which that identification—called “classifying” in the ’250 patent—takes place. (See Petition 48.)

104. Petitioner also argues that a POSITA would have sought to incorporate “Kirby’s teachings on classifying as malware objects that ‘touch’ or have interacted with known malware.” (Petition 47.) But, again, Petitioner does not explain how or where a POSITA would have incorporated those teachings—not least because Petitioner does not propose the use of Kirby in its combination as reading on the “classifying” limitations. Further, because Halperin’s own approach is directed to collating occurrences of suspicious network-level behaviors instead of being directed to identifying particular pieces of malware, in my opinion a POSITA would not have had any way to add Kirby’s “touch” teaching into a Halperin-based system. Identifying objects that interact with malware requires at a minimum that the system first identify which objects are malware so that it can identify which other objects have interacted with those objects, but in Halperin’s aggregate-behavior detection system, no such first identification takes place.

105. As is the case with Mankins, Halperin would also discourage a POSITA from turning to Kirby to fill any gaps that POSITA identified in Halperin. As discussed above, Halperin discourages the use of “software that employs heuristic techniques” that rely on identifying acceptable behaviors and deviations from those behaviors to identify software as malware. Kirby teaches a “behavioral heuristic” that “identifies actions known to be common to undesirable activity.” (EX-1007 3:7-9). Moreover, because Kirby teaches using two different kinds of malware

detection, it would be more—not less—likely to generate false alarms, which Halperin considers an undesirable side effect of heuristic-based software. (Ex. 1006 ¶ 6.)

106. I am informed that in addition to proposing a motivation for combining references, a petitioner has a second responsibility—to show that the POSITA would have a reasonable expectation that he would succeed in combining the prior art in order to achieve the invention claimed in the challenged patent. But I see nothing in the petition that suggests any POSITA would have expected success in combining Kirby with Halperin and Mankins, either singly or in combination. In my opinion, a POSITA would not have expected success in combining these three references, because Halperin takes a significantly different, aggregated-behavior approach to malware detection from the more object-centric approaches in Kirby and Mankins. In my opinion, the only way the Petition suggests to combine the Kirby and Mankins references with Halperin is to use the '250 patent as a template or guide in assembling a finished product from unrelated parts of the three references. I am informed that this is an impermissible basis upon which to challenge the '250 patent's patentability.

C. Independent Claims 1, 13, and 25

107. The Petitioner relies on the same analysis of Halperin to support Independent Limitations [1P] to [1C], [13P] to [13C], and [25P] to [25B], and [25I],

as well as for dependent claims 2 and 14 in both Grounds 1 and 2. (Petition 24-29, 42-43, 48, 50, 60-62) As I discussed above, I disagree with Dr. Newman that there is a motivation to combine Halperin with Mankins. The two references employ disparate detection techniques to target different types of malware problems resulting in different ameliorative steps. Where Halperin is designed to detect viral outbreaks in a network by analyzing and correlating computer behavior, Mankins is designed to prevent system intrusions by analyzing and blocking certain system calls. As a result of their differences, neither would benefit the other, their combination would require significant modification to Halperin's system, and their combination would suffer from inefficiencies.

108. I further disagree with Dr. Newman that Halperin alone teaches each of [1P] to [1C]. *First*, I disagree that Halperin receives data *about an object* from remote computers. It is my opinion that Halperin, instead, receives only behavior data about actions generically performed by a computer—without regard to any specific object. *Second*, I do not agree that Halperin utilizes received data to determine that a computer object is malware. *Third*, I do not agree that Halperin *classifies objects* as malware. Additionally, I disagree with Dr. Newman that Halperin, even in combination with Mankins, teaches [1E] nor would a POSITA have been motivated to make that combination.

1. **Halperin Does Not Disclose Limitations [1P] to [1C], [13A] to [13C], [25P] to [25B], and [25I]**

109. It is my opinion that Halperin does not teach or render obvious the limitations directed at receiving information about an object from remote computers and using that information to determine whether the object is malware and classifying the object.

110. Halperin is directed to determining whether there is a viral outbreak in a network by examining when a percentage of computers perform the same suspicious behavior, such as, for example, sending emails to too many users within a predetermined amount of time, sending emails without interaction from a human user, or deleting too many files within a predefined period of time. (EX-1006 ¶¶48-56.) These behaviors are not necessarily performed by any specific object and so Halperin's data collection has no need to, and indeed does not, identify that information with reference to any particular object. As such, Halperin also does not rely on information about a particular object to reach a determination and classify that object as malware.

- a. **[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.**

111. I disagree with Dr. Newman that Halperin discloses this limitation. I further do not read Halperin to teach that the data that is communicated to Halperin's

base computer is object-specific or even associated with any particular object. In other words, I do not understand Halperin's detections to be focused on any particular object and so I do not understand Halperin to communicate any data that is "about a computer object."

112. In his description of the overview of the '250 patent, Dr. Newman points out that the '250 patent "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." (EX-1004 ¶ 43 (*citing* EX-1001 10:41-47).) In contrast, Halperin does not gather any of this type of data. Instead, Halperin gathers computer-related data that is not necessarily tied to any object. Halperin operates using sensors that monitor computer behavior for certain patterns of suspicious *actions*, that "trip" or "trigger" an alert to the central computer when the suspicious behavior is detected. (EX-1006 [0046].) During Halperin's detection phase, "the system logs every event of a sensor identifying a suspicious behavior on a specific computer into a central database ... preferably includ[ing] the type of event, the time it occurred and on which computer it occurred (if relevant)." (EX-1006 [0090].) Halperin does not disclose that the sensors collect any object-specific information. To the contrary, Halperin specifically identifies the types of information its system stores in the central database, none of which is object-specific.

113. Further, Halperin sets forth a list of computer behaviors that form the basis of its behavior-pattern-based detection. (See EX-1006 [0039]-[0046].) Those detections include behaviors that would not necessarily be correlated to any specific computer object, for example, “[s]ending messages to more than a predefined number of users ...;” “[s]ending messages not as a result of a direct user interaction with a Graphic User Interface ...;” “[m]odifying operating system files ...;” “[d]eleting more than a predefined number of files.” (EX-1006 [0039]–[0046].) Because none of these behaviors are tied to an object, it cannot be presumed that Halperin would even inherently communicate data about an object.

114. Dr. Newman suggests that “[a] POSITA would have known that ‘target behavior’ and ‘predefined suspicious event[s]’ are data about a computer object like a virus.” (EX-1004 ¶ 77.) But I disagree that these target behaviors are necessarily tied to any one object’s behavior. For example, a user—who is not an object according to the definition in the ’250 Patent—could “delet[e] more than a predefined number of files” or “modify[] operating system files” or “send[] messages to more than a predefined number of users.” Similarly, computers are able to detect behaviors that, though not necessarily directed by a user, are not tied to any particular object. For example, a system according to Halperin could detect whether remote computers are connected to the Internet or how much hard drive space remains free on each remote computer or what peripherals (*e.g.*, graphics cards)

might be attached to each remote computer. Halperin's detections are very high level and, as such, gathering object-specific behavior would require more processing than Halperin's method and system requires.

115. Accordingly, Halperin does not disclose, either expressly or inherently, receiving data about a computer object at a base computer.

- b. [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;**

116. For reasons similar to those I set forth for [1A], and which I incorporate by reference here, I disagree with Dr. Newman that Halperin teaches "determining in the base computer whether the data about the computer object received from the plural computers indicates that the computer object is malware." Again, I understand Halperin to perform an object-agnostic determination of whether there are symptoms of malware activity across a computer network.

117. Dr. Newman proposes that because Halperin "collect[s] information regarding target behavior" and "determine[s] whether the correlated target behavior corresponds to a predefined suspicious behavior pattern of a target behavior as an indication that the computer virus may have infected those computers" that Halperin discloses making determinations about specific computer objects. (EX-1004 ¶ 79 (citing EX-1006 [0047]).) Dr. Newman does not explain why correlating target behavior corresponds to determining that a specific object is malware and I disagree

that any such explanation exists. As discussed in regards to limitation [1A], Halperin does not collect object-specific information and thus is incapable of making object-specific determinations. Even more so, correlating behavior in no way implies that the actor of that behavior is identified. For example, it is entirely possible to determine that “a certain percentage of computers in the network delet[ed] a file by the same name within the last 1 hour” without knowing which objects on those computers was responsible for deleting that file. (EX-1006 [0051].) Namely, even if Halperin did collect object-specific information (which it does not), Dr. Newman’s argument would fail.

118. Accordingly, I do not read Halperin to communicate data about objects to the central computer. Therefore, I further do not read Halperin to make a determination about whether a computer object is malware based on “data about the computer object.”

c. [1C] classifying the computer object as malware when the data indicates that the computer object is malware

119. For reasons similar to those I set forth for [1A], and [1B], and which I incorporate by reference here, I disagree with Dr. Newman that Halperin teaches “classifying the computer object as malware when the data indicates that the computer object is malware.” I disagree that Halperin classifies *any* malware, let alone that it classifies a particular computer object as malware.

120. Halperin employs an aggregated-behavior-based approach to malware detection that does not tie behaviors to any specific object. In other words, Halperin detects malware *symptoms*. The '250 patent (including the preamble to claims 1 and 13) instead employs an object-centric system: it claims an invention in which the system “classif[ies]” a particular “*computer object* as malware” rather than merely inferring the possible presence of *some* malware somewhere on a network from the parallel behavior of multiple computers on that network.

121. A simple example illustrates the distinction between these two approaches. The Transportation Security Administration sometimes employs trained dogs at airports to find illicit or dangerous substances stored in travelers' luggage. The dog might be trained to alert when it smells a range of illicit substances, including multiple types of illegal drugs and explosives. A dog alerting on a traveler's luggage might trigger authorities to detain the traveler for further examination. That is analogous to how Halperin's system operates: it identifies symptoms of malware (here, the scent of *some as-yet unidentified* illicit substance in the traveler's luggage) and takes preventative action such as shutting down network traffic.

122. What Halperin does not teach is how to perform the next step: identifying what in the traveler's luggage triggered the alert—one or another type of drug or explosive, or perhaps a false positive resulting from non-illicit substances

that may smell like illicit substances, or perhaps even a dog treat that may cause the dog to overcome its training and issue a “false positive.” Detaining the traveler may improve public safety, but nothing about that initial alert or traveler detention actually classifies the object in the traveler’s luggage that is responsible for the alert.

123. Nothing in Halperin teaches a POSITA how to classify any particular piece of software on a central computer or any remote computer as the culprit in generating the symptoms Halperin is designed to aggregate. In my opinion, therefore, Halperin does not teach classifying any computer object as malware.

124. In support of its argument that Halperin discloses this limitation, Petitioner cites an instance in which Halperin refers to “identify[ing] malicious code.” (Petition 28, citing Ex. 1006 ¶ 87.) But in my opinion, that is an example of Halperin’s drafter using imprecise description, because Halperin does not teach identifying any code as malicious. The paragraph Petitioner references merely refers back to the use of Halperin’s *behavior* “sensors” to suggest “the existence of a virus”—not to classify any particular piece of software as a virus. (See Ex. 1006 ¶ 87.) In fact, nothing in this paragraph suggests that Halperin itself performs any identification. Halperin explains that the system requires “retrospective external inputs (from an operator or standard antivirus software).” (See Ex. 1006 ¶ 87.) A POSITA would understand that the word “external” means that any retrospective inputs are not part of Halperin’s system. As I explain above, in [1P] through [1B],

Halperin’s sensors can detect suspicious behavior in the aggregate in the network but are not capable of identifying malicious code.

125. Similarly, Petitioner relies on another passage of Halperin that refers to “the identification of virus activity.” (Petition 28, citing Ex. 1006 ¶ 61). Unlike the imprecise description in ¶ 87, “identification of virus *activity*” is an accurate description of what Halperin does—but it is not the same as classifying any object as malware. A detective surveying a crime scene may be able to identify criminal *activity* (such as by observing bullet holes in a wall, blood on a carpet, and a dead body) but that detective has a significant additional amount of work to do before he can classify any individual as the *criminal* responsible for that activity.

126. At the outset, Halperin does not classify any malware, it simply detects symptoms of viral outbreaks. Halperin explains that it correlates “target behavior ... to a predefined suspicious behavior pattern of target behavior as an indication that a computer virus may have infected those computers.” (EX-1006 [0047].) Identifying indications that a computer virus may have infected several computers is not classifying malware, because it merely identifies symptoms of malware and deduces that malware must be present. By way of analogy, Halperin’s disclosure is akin to suspecting that a pandemic exists because doctors all over the world have found that millions of people all over the world are simultaneously exhibiting signs of illness (shortness of breath, fevers, etc.). Classifying malware is akin to a determination by

one doctor that one individual has COVID-19. The two scenarios are not dependent on each other: a doctor can diagnose an individual patient regardless of the existence of a pandemic and it is possible to suspect that a pandemic exists without knowing the specific virus responsible for it. Similarly, Halperin deduces that a virus is likely responsible for the suspicious symptoms, but does not disclose detecting the specific computer object responsible for creating those suspicious symptoms (actually classifying the computer object).

127. In my opinion, Halperin’s silence on identifying the particular piece of software responsible for malware activity on a network is a feature of Halperin’s design. Halperin’s malware-response teachings—“containment actions”—are designed to be employed rapidly and without any knowledge about which particular piece of software is responsible for the symptoms Halperin’s system observes. (See Ex. 1006 ¶¶ 23-30.) For example, Halperin teaches halting all traffic on a network in response to observing parallel activity on multiple computers that indicates the presence of malware. (Ex. 1006 ¶ 28.) Although that is a blunt-force response, it may be employed quite rapidly and will necessarily halt further spread of the malware among computers on the network. Awaiting identification of the particular piece of software that caused the symptoms Halperin’s sensors have observed would delay this rapid response, potentially allowing further spread of the malware on the network.

128. Dr. Newman states that “Halperin compares the suspect object to predefined criteria.” (EX-1004 ¶ 81.) I disagree that Halperin does so and Dr. Newman provides no specific explanation as to how the six cited disparate paragraphs of Halperin support his assertion. It is my opinion that none of them do. As previously discussed in [1A] and [1B], Halperin receives no information about any specific computer objects and thus is unable to compare any suspect objects to predefined criteria. Instead, Halperin receives information about computer behavior, which it compares to predefined criteria. The two citations that Dr. Newman does draw attention to are inapposite to his conclusion. At [0087], Halperin discusses the “learning” phase of the system. (EX-1006 [0087].) In this phase, the system runs and receives “retrospective external inputs (from an operator or standard antivirus software) on when there was a virus in the system.” (*Id.*) At no point does Halperin indicate that the system itself can identify specific objects as viruses. At [0061], Halperin is explicit that it identifies suspicious *activity* and that the suspicious activity must be identified as a virus by a system operator. (EX-1006 [0061].) This means that activity classified. At no point does Halperin indicate that a specific computer object is classified. All of Dr. Newman’s other cites suffer from similar mischaracterizations.

129. Dr. Newman’s second conclusion, that because Halperin activates a virus alarm or initiates virus containment actions it must have classified an object as

malware, is simply nonsensical. Halperin explains that it initiates those responses because a certain threshold of predefined suspicious actions in the *entire system* was met. (EX-1006 [0090].) This is in no way explicitly correlated to a specific computer object. First, this paragraph explains that the alarms are determined based on a “predefined threshold.” (*Id.*) It is illogical to say that a computer object has a threshold. Instead, what does have a threshold is a probability that suspicious actions indicate that there is in fact a “virus in the system.” (*Id.*) Second, as discussed previously, Halperin does not collect object-specific behavior and thus cannot initiate a virus alarm based on object-specific classifications.

130. To reinforce this point, I offer another analogy. When a patient calls his doctor and says he is experiencing coughing, sneezing, and a runny nose, the doctor may recommend, among other things, that the patient stay home from work or school to avoid passing on the presumed infection to others. But the fact that the doctor detected symptoms and took containment actions does not mean that he also necessarily diagnosed (or classified) the patient as having any virus—much less that the doctor identified the particular virus responsible for the patient’s symptoms.

131. Similarly, here, when Halperin detects malware symptoms, it initiates actions to prevent further spread from potentially infected systems. But Halperin’s disclosure of actions taken to contain a possible virus on a network does not inherently disclose or indicate to a POSITA that Halperin has classified any

particular object as malware—particularly where none of the remedial containment actions Halperin discloses requires knowledge that a particular object might be or is in fact malware. (See EX-1006 [0048]-[0056].)

132. Accordingly, I do not read Halperin to teach classifying any computer objects as malware.

2. The Petition’s Combination of Halperin in View of Mankins Does Not Render Obvious Limitations [1D] to [1I], [13D] to [13I], [25D], [25F], [25G], [25I] and [25J] Nor Dependent Limitations [2], [14] and [25B].

133. As I discussed above in section VI.A I disagree with Dr. Newman that there is a motivation to combine Halperin and Mankins to perform this limitation. The two references employ disparate detection; where Halperin is designed to detect symptoms of viral outbreaks in a network by analyzing and correlating computer behavior and to contain those potential outbreaks, Mankins is designed to prevent system intrusions by analyzing and blocking certain system calls within the operating system kernel. As a result of their differences, neither would benefit the other, their combination would suffer from inefficiencies, and adding Mankins would at impair Halperin’s ability to perform its own disclosed detection methodology.

a. [1D] initially classifying the computer object as not malware.

134. Petitioner argues that Halperin and Mankins each individually disclose this limitation. I disagree. Halperin does not disclose performing an initial

classification of a computer object as not malware. Petitioner is clear about its argument: any time a computer object is *not affirmatively* classified as malware, it is classified as “not malware.” (Petition 29.) This is not true. For the ’250 patent, the purpose of the classification is to allow the system to identify programs that are pre-authorized to run and determine whether they should be monitored. (See EX-1001 Fig. 3.) Petitioner’s argument that the converse of a classification of malware is a classification of not malware is flawed because it does not enable subsequent recognition of which programs are running at that time. Halperin’s continuous scanning when it does not detect any suspicious target behavior is not an affirmative classification that all programs running at that time are “not malware.”

135. Likewise, Mankins does not disclose this limitation. Petitioner relies on Mankins’s disclosure of a training phase to learn a program’s normal behavior as Mankins performing an initial classification of objects as not malware. (Petition 29-30.) A training phase cannot disclose “initially classifying the object as not malware” because it misses a critical component of the claim limitation: the classification must be performed based upon a “determining” that “does not indicate that the computer object is malware.” In Mankins, objects during a training phase are assumed to be not malware so that their normal behavior can be learned. Mankins is explicit that this learning takes place when the programs are not “subject to intrusion attacks.” (EX-1008 6:63.) Thus, the Mankins learning phase is predicated on the computer

programs already being known as not malware. Because Mankins does not perform a classification based upon any “determining,” it cannot disclose this limitation.

- b. [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.**

136. For this limitation, Dr. Newman first suggests in conclusory fashion that “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.” (EX-1004 ¶ 91.) As discussed in Section VI.A.2, a POSITA would not have been motivated to implement this suggested configuration because it is not a true combination and runs contrary to Halperin’s teachings. (*Supra* ¶80.) As I’ve described throughout, there are many differences that make combining Halperin and Mankins challenging or ineffective, including the type of information each system analyzes.

137. Dr. Newman’s next suggestion is that Mankins could be implemented “in a distributed fashion, including through the use of Halperin’s system architecture.” (EX-1004 ¶ 92.) As discussed in Section VI.A.2, a POSITA would not have been motivated to implement this suggested configuration because it is premised on false assumptions and runs contrary to the core teachings of Mankins. (*Supra* ¶75-77.) Moreover, contrary to Dr. Newman’s bare conclusion, the mere fact

that Halperin and Mankins may both operate according to a distributed network would not be motivation enough to overcome the issues with their combination.

138. For the same reasons he provided for in his primary motivation-to-combine section, Dr. Newman next suggests that a POSITA would have contemplated combining Halperin with Mankins such that Mankins's Bloom filters would be added to or substituted in for Halperin's sensors. (EX-1004 ¶ 94.) I disagree. As I've explained, Halperin teaches a high-level approach to detecting symptoms of viral activity in a network by determining when a threshold number of computers perform suspicious behaviors—for example sending emails to too many addresses within a predetermined time frame. These detections are object-agnostic. There is no indication that Halperin's system would be in any way improved by collecting system-call behavior. Adding Mankins's Bloom filters to Halperin's analysis would be inefficient, as it would communicate far more information than Halperin's behavior-pattern analysis requires. Moreover, replacing Halperin's sensors with Mankins's Bloom filters would require rewriting Halperin's correlation engine.

139. Furthermore, I do not understand this motivation-to-combine argument to be relevant to limitation [1E]. If a POSITA were to combine Halperin with Mankins such that Mankins's Bloom filters form the basis of data sent to Halperin's base computer, that is relevant to limitations [1A]-[1C]—the limitations directed at

gathering data from remote computers at a base computer. Despite this, the Petitioner never mapped this combination to those limitations. So even if this combination were at all plausible (which it is not), the motivation Dr. Newman suggests would not support combining Halperin and Mankins to arrive at limitation [1E].

140. In addition to this motivation-to-combine theory, Dr. Newman also suggests that combining Halperin with Mankins would reduce false positives by “improving the robustness of the ‘normal’ behavior data.” The only support Dr. Newman relies on for this position is the fact that both Halperin and Mankins rely on a distributed system to reduce false positives. (EX-1004 ¶ 96.) As I discuss extensively in Section VI.A.3, Dr. Newman does not explain how Mankins’s teaching of reducing false positives by generating Bloom filters at remote devices would have motivated a POSITA to combine Mankins with Halperin, especially when Halperin already reduces false positives by operating in a distributed environment. Indeed, Halperin teaches adjusting its sensors “retrospectively once a series of system alerts have been identified as either real virus alerts or false alarms.” (EX-1006 [0075].) In other words, Dr. Newman has advocated adding Mankins to Halperin to accomplish something Halperin already teaches. After reading Halperin, a POSITA would not be motivated to search for a solution to a problem Halperin already purports to address.

141. In my opinion a POSITA reading Halperin would have had no motivation to consider additionally adding Mankins to “automatically generat[e] a mask for [a] 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 step necessary.” Object-specific detections are outside the scope of Halperin, and Halperin even teaches away from the use of normal-behavior based heuristics. (*See* EX-1006 [0006].) Presumably, this is why Dr. Newman advanced his theory about replacing Halperin’s sensors with Mankins’s Bloom filters. But as I set forth above, this combination is irrelevant to limitation [1E] and would not even be effective for Halperin.

142. Finally, Dr. Newman posits that a POSITA would have had “a reasonable expectation of success in modifying Halperin to incorporate Mankins’s Bloom filter data object anomaly detection approach” because “anomaly detection was a known approach to virus detection.” (EX-1004 ¶ 97.) However, a POSITA would not have had a reasonable expectation of success because a POSITA would not be able to reconcile the fundamental difference of the levels on which Halperin and Mankins operate (at the computer activity level and the object level, respectively). At this step of the proposed combined method, Halperin has not identified any suspicious behavior. Thus, a POSITA would not know how to focus Mankins’s object-specific detection methods. Because Halperin has not provided

any information to help narrow down even the suspected computer, the only feasible solution is to run Mankins on every computer in the network, which means creating and monitoring a Bloom filter for every object on those computers. This is a resource-intensive and unwieldy endeavor.

143. For at least these reasons, it is my opinion that a POSITA would not have combined Halperin with Mankins—especially not to arrive at limitation [1E].

c. [1I]: reclassifying the computer object as malware when the actual monitored behavior extends beyond the mask.

144. Neither Halperin nor Mankins discloses the step of “reclassifying the computer object as malware.” The ’250 patent recognizes that a system based entirely on signature matching cannot continually assess whether objects are safe or not. (EX-1001 3:8-18.) Thus, the purpose of the reclassification step in the ’250 patent is to ensure that the system remains dynamic and evolves in order to remain constantly vigilant for new malware. By allowing the ’250 patent to analyze an object more than once, and produce differing classifications of the object based on its “actual monitored behavior,” the reclassification step provides a security feature not present in the prior art. Neither Halperin nor Mankins disclose this limitation because neither Halperin nor Mankins perform a secondary classification.

145. The “virus containment actions” Petitioner cites do not constitute a reclassification. First, as I explained above, because Halperin never performs an

initial classification, it cannot perform a reclassification. Secondly, none of the seven “virus containment actions” Halperin discloses require or even contemplate identifying a specific computer object. The actions described in Halperin are all object-agnostic because they are taken at a network level. Halperin’s actions, such as “suspending messages sent by a computer,” “notifying a user,” or “notifying a system administrator” are not tied to a specific object. Neither Petitioner nor Dr. Newman explain how any of these actions constitute a reclassification of *the* computer object described in the ’250 patent.

146. I also disagree with Dr. Newman that Mankins renders limitation [1I] obvious. Dr. Newman points to two of Mankins’s teachings, which he believes disclose “reclassifying the computer object as malware when the actual monitored behavior extends beyond the mask.” First, Dr. Newman points to Mankins’s teachings that “[i]f a match is not found, the observed system-call sequence is determined to be anomalous and may indicate an intrusion attempt.” (EX-1004 ¶ 110 (citing Ex. 1008 at 6:39-41).) But this passage does not suggest a classification of any particular computer object. Instead, a series of system calls is identified as suspicious. Second, Dr. Newman points to Mankins’s “predetermined actions designed to protect the computer.” (EX-1004 ¶ 110 (citing Ex. 1008, 11:8-11).) But these predetermined actions do not include classifying an object as malware, much less reclassifying it. Mankins does not elaborate on what “predetermined actions

designed to protect the computer” may be. Like in Halperin, the predetermined actions may be *preventative* as opposed to a specific classifying action. Further, Mankins does not suggest—and nothing requires—that the responsive action be specific to the program whose system call Mankins’s Bloom filter has flagged as suspicious. For example, Mankins could take a similar approach to Halperin and notify a user or a system administrator of the anomalous system call behavior. As I described above, this kind of preventative action is not a reclassification.

147. Regardless, I do not agree that it is obvious that “predetermined actions” would include reclassifying the program that performed the system-call sequence. Because Mankins does not elaborate on what such “predetermined actions” may be, they are left to speculation and common sense. A POSITA would not recognize that reclassifying (implying that the program has already been classified before) is an obvious response to triggering the Bloom filter data object. Immediately castigating a program merely for performing an unrecognized system call is an extreme response – one that would often at least be validated by a human.

148. For these reasons, it is my opinion that Mankins does not teach or render obvious “reclassifying the computer object as malware when the actual monitored behaviour extends beyond that permitted by the mask.”

- d. [2], [14], [25B] 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 object.**

149. Halperin alone does not disclose including the current name of any computer object when sending data from the remote computers to the base computer. A POSITA would have understood that “the computer object” to which these limitations refer is the same computer object in respective independent limitations [1A] and [13A], based on antecedent basis. Accordingly, “the current name of the object” must refer to the same object whose behavior data was received at the base computer. Neither Petitioner nor Dr. Newman explain where Halperin discloses this.

150. It would be illogical to assume that the filename of an attachment sent in a message is identical to the filename of the computer object responsible for sending it. Computer malware is designed to evade detection. It does so by using innocuous filenames, which entice the recipients of message attachments to open the attachment without doubting its legitimacy. Once the user opens the file attachment, perhaps named “resume.pdf” or “client_invoice.xlsx,” the attachment executes malicious code to download the malware onto the user’s machine. At some point in the future, the malware will attempt to propagate itself by sending more messages to other machines in the network. Halperin explains this exact scenario in paragraph 31 when it describes that a computer may “become infected by a computer virus that

generates its own messages, as the virus attempts to propagate” and “automatically generates messages that incorporate the virus.” (See EX-1006 ¶ 31.)

151. Because of the difference between the virus and its propagation mechanism, a POSITA would have understood that the virus object responsible for sending the file attachments and the file attachments themselves are two different objects. Thus, Petitioner and Dr. Newman are incorrect to assert that Halperin’s detection of multiple “computers in the network sending the same file attachment via a message in the last 15 minutes” discloses sending “the current name of the object” to the base computer as contemplated in claims 2 and 14. That teaching refers to the file attachments themselves and not the malware object responsible for sending said attachments.

VII. CONCLUSION

152. For at least the foregoing reasons, it is my opinion that Petitioner’s grounds do not render obvious any of the challenged claims of the ’250 Patent.

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

Executed this 2nd day of January, 2024 in Los Angeles.

Respectfully submitted,



Nenad Medvidovic, Ph.D.