

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AO KASPERKY LAB,

Petitioner

v.

OPEN TEXT INC.

Patent Owner

Case IPR2023-00556

Patent No. 8,726,389

**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 '389 Patent.....	11
	A. Summary Of The '389 Patent.....	11
	B. Claim Construction	19
	C. Level Of Ordinary Skill In The Art.....	20
V.	OVERVIEW OF PRIOR ART	21
	A. Overview Of Kester.....	21
	B. Overview Of Kennedy.....	34
	C. Overview Of Honig	38
VI.	KESTER IN VIEW OF Kennedy and Honig DOES NOT DISCLOSE OR RENDER OBVIOUS THE CHALLENGED CLAIMS.....	39
VII.	Ground 1: Kester in view of Kennedy did not render claims 1-13, 15- 27, and 29-30 obvious.	39
	A. Neither Kester nor Kennedy teaches that identifying a relationship between two objects includes comparing those objects, as required by the independent claims.....	40
	1. Kester does not teach the claimed “comparing.”	42
	2. Kennedy does not teach the claimed “comparing.”	44
	B. Petitioner fails to show that the proposed combination would meet the claimed “receiving,” “correlating,” and “comparing” steps, as required by the independent claims.	47

1.	Petitioner’s combination does not teach or suggest “receiving” the claimed “information about events” from remote computers, as required by independent claims 1 and 15.....	48
2.	The proposed combination does not teach or suggest “correlating” event data from Kennedy’s MSDM.....	50
3.	The proposed combination does not teach or suggest “comparing ... correlated data ... to other objects or entities to identify relationships between the correlated data and the other objects or entities,” as required by independent claims 1 and 15.....	52
C.	Petitioner fails to show that the proposed combination would have rendered the dependent claims obvious.....	55
1.	Petitioner’s analysis of dependent claims 2-7 and 16-21 is incompatible with its analysis of the independent claims.	56
2.	Petitioner has not shown that the data used for the “comparing” includes any of the information recited in dependent claims 8 and 22.....	59
VIII.	Ground 2: Kester in view of Kennedy and Honig did not render claims 14 and 28 obvious.....	60
IX.	CLAIMS AND/OR LIMITATIONS NOT ADDRESSED IN THIS DECLARATION	69
X.	SECONDARY OBJECTIVE INDICIA OF NONOBVIOUSNESS	69
XI.	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>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)
2004	<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)
2005	<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)
2006	<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)
2007	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Dec. 9, 2022 Transcript of Motions Hearing, Dkt. 133 (W.D. Tex. Dec. 15, 2022) (excerpted)
2008	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
2009	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,

Exhibit	Description
	2023)”, available https://www.uscourts.gov/sites/default/files/data_tables/fcms_na_distprofile0331.2023.pdf
2010	<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)
2011	<i>Webroot, Inc. and Open Text Inc., v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.</i> , Case No. 6:22-cv-00241-ADA-DTG, First Amended Complaint for Patent Infringement, Dkt. 60 (W.D. Tex., Sept. 8, 2022) (excerpted)
2012	<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)
2013	<i>Webroot, Inc. and Open Text Inc., v. ForcePoint LLC</i> , Case No. 6:22-cv-00342-ADA-DTG, Defendant ForcePoint LLC’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2014	<i>Webroot, Inc. and Open Text Inc., v. Sophos Ltd.</i> , Case No. 6:22-cv-00240-ADA-DTG, Defendant Sophos Ltd.’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2015	<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)
2016	<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)
2017	<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)

Exhibit	Description
2018	<i>Ex Parte</i> James M. Brennan, et al., Appeal 2021-003606, Application 15/285,875, Decision On Appeal Statement of the Case (PTAB Jul. 26, 2022)
2019	File History of U.S. Patent No. 8,418,250 (“250 File History”)
2020	Declaration of Professor Nenad Medividovic, Ph.D. In Support of Patent Owner’s Response, dated March 29, 2024
2021	<i>AO Kaspersky Lab v. Open Text Inc.</i> , Case No. IPR2023-00556, Deposition Transcript of Wenke Lee, Ph.D., dated March 15, 2024
2022	<i>Webroot, Inc. and Open Text Inc. v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA-DTG, Opening Claim Construction Brief From: [AO Kaspersky Lab], Dkt. 86 (E.D. Tex. Oct. 28, 2022) (excerpted)

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 the petition for Inter Partes Review (“Petition”) of U.S. Patent 8,726,389 (“the ’389 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 ’389 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 ’389 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. 8,726,389 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 ’389 patent at the time of the alleged invention. I have also been asked to provide my

technical opinions on how concepts in the '389 patent specification relate to claim limitations of the '389 patent. In reaching the opinions provided herein, I have considered the '389 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 filed, 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 more than 25 years, my research has focused on the building blocks that comprise software systems: software components (computational and data

elements) and connectors (interaction-facilitating elements), and mechanisms for their implementation, primarily middleware facilities and software frameworks. In particular, 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 such 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 currently hold the title of Professor with tenure. Since July 2022, I have served as my department’s Chairman. Between January 2009 and January 2013, I served as the Director of the Center for Systems and Software Engineering at USC. 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 19 Ph.D. students and advise 7 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 different 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 of work. 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 one 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. I have co-authored three patents in the area of cybersecurity.

21. I have been named a Distinguished Scientist of the ACM. I have been elected a Fellow of the IEEE, IEEE's highest grade that is granted to less than 0.1% of its membership annually.

II. UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES

22. I am not a lawyer, and I will not provide any legal opinions. Although I am not a lawyer, I have been advised certain legal standards are to be applied by technical experts in forming opinions regarding the meaning and validity of patent claims.

23. 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.

24. 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.

25. 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 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;
 - “Obvious to try” – choosing from a finite number of identified, predictable solutions with a reasonable expectation of success;
 - 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.

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

27. I have been informed that a prior art reference may be relied upon not only for what it expressly teaches, but also for what is inherent therein—i.e., what necessarily flows from the express teachings. I further understand that the mere fact that a certain thing may result from a given set of circumstances is not sufficient.

III. STATE OF THE PRIOR ART

28. In the field of computer security, “malware” is an executable or other computer object that contains malicious code. (EX-1001 1:19-22.) Examples include viruses, trojans, worms, etc. (*Id.*)

29. A traditional approach to detecting and blocking malware is to scan computer files for “signatures” associated with known malware. (EX-1001 1:23-2:23.) Using this approach, however, when new malware is introduced, service providers must identify the new malware, analyze it to find new signatures, and then

update anti-malware programs to search for those new signatures in computer objects. (*Id.*)

30. Another traditional approach leverages the insight that files that have been around longer without behaving badly may be safe. For example, an anti-malware program could check databases of known files to determine whether a given file has been known long enough to be deemed safe. (EX-1001 1:48-52.)

31. A major shortcoming of these traditional approaches was their inability to detect new malware quickly. The systems were reactive, in that they waited for malware to present itself (often by infecting networks and computers) before taking any action, or instead, waiting around for programs to age sufficiently without incident. (EX-1001 1:35-42, 1:53-59.)

32. To quicken these response times, some “community-based” approaches were developed, where a “central computer” gathered and used information from multiple “local” computers connected to it over a computer network. (EX-1001 1:65-2:9.) But those approaches had several shortcomings, including that they required sending large files to the central computer and moreover, that they still relied largely on manual analyses of those files. (EX-1001 2:9-17, 2:63-3:3.) Such manual analyses of the actual file was slow, sometimes “tak[ing] days.”

33. Accordingly, there was a long-felt need for an improved way to quickly determine whether files were safe or malware.

IV. THE '389 PATENT

A. Summary Of The '389 Patent

34. The '389 patent introduced improvements in malware detection that overcame these problems in the prior art. (EX-1001 3:14-24, Cls. 1, 15.) Rather than “rel[y] solely on signature matching” and “detailed analysis of the object itself” as in the prior art, the claimed system relies on “sophisticated pattern analysis” of data about “the objects and/or their effects on the different remote computers.” (EX-1001 3:14-24).

35. The claimed system utilizes a community-based approach in which “remote computers” send to a centralized “base computer” certain data concerning computer objects stored on those “remote computers.” (*Id.* 3:3-13.). In a preferred embodiment, “the data [received about each object] is sent in the form of a key” with three severable components: (1) a first component “that represents executable instructions contained within or constituted about the object,” (2) a second component “that represents data about said object,” such as object metadata (*e.g.*, “names, locations, vendor, product and version”) and/or event metadata (*e.g.*, “events initiated by or involving the object when the object is created, configured or

runs”), and (3) a third component “that represents the physical size of the object.” (*Id.* 3:48-4:26, 9:49-54).

36. The data may be “captured and corelated from all of the remote computers that have seen [an] object.” (*Id.* 10:43-11:4.) Two objects are regarded as “the same” if they have sufficient overlap in the first and third components—i.e., sufficiently similar executable instructions and size. (*Id.* 3:57-4:12, 9:66-10:34.)

37. “[T]o establish if an object is safe, unsafe or suspicious” the “base computer” applies various disclosed “heuristics” to the collected data. (*Id.* 10:41-42.) In one of those heuristics, the system can “utilize[] an object’s relationship to other objects that act upon it or upon which it acts.” (*Id.* 11:5-60.) “[A]n object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object.” (*Id.* 11:11-13). Example relationships may include “which object created this object, which objects this object created, which objects created a registry key to run or configure this object, which objects were configured by or had registry keys created by this object, etc.” (*Id.* 11:7-11.)

38. In some embodiments, relationships may also depend on whether two objects share certain “genes.” (*Id.* 17:22-18:15.) For example, “[t]he key relating to [a] first object may be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object.” (*Id.* 17:22-30.)

39. The patent discloses various classification techniques that can be applied using these gene-based family relationships. (*Id.* 17:30-18:15.) For instance, a child process that inherited certain genes may also inherit its parent's classification, and if the parent's or child's subsequent behavior reveals it is malware, the other (and all siblings) can be marked as malware too. (*Id.* 17:43-51.) Similarly, the system can monitor "twins of objects (e.g., the same process running on other systems in the community) to compare the pattern of their issue and to make decisions as to whether or not to treat any particular process as malware." (*Id.* 18:3-7.) Thus, the invention can identify useful family relationships based on shared "genes."

40. By collecting relationship data about objects from many computers, the claimed system can create "a complex series of correlation points, allowing ancestral relationships to be considered for any object ... by event." (*Id.* 11:13-17.) "Objects that have specific relationships, volumes of relationships or mixes of relationships to one type or another may be judged safe or unsafe accordingly." (*Id.* 11:21-24.) And because "a typical implementation in the field" may include "several thousands or even millions of remote computers connected or connectable to the base computer," "any objects that are newly released upon the Internet or that otherwise are found on any of these remote computers will soon be found and signatures

created and sent to the base computer by the respective remote computers.” (*Id.* 8:56-63.)

41. The patent also discloses the idea for monitoring the behavior of objects by generating a “mask” that defines the object’s “acceptable behavior,” monitoring the object’s operation, and reclassifying the object as malware if the monitored behavior extends beyond the behavior the mask permits. (*Id.* 4:27-36.)

42. The independent claims (claims 1 and 15) recite inventive aspects of this invention, including the ideas of a “base computer” that (1) receives data “about” an object (including “event” data) from multiple “remote computers,” (2) correlates the data received from the remote computers, (3) compares the correlated data to other objects to identify “relationships” to those other objects, and (4) classifies the object based on that comparison.

Claim 1 recites:

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

[1.1] at a base computer, receiving data about a computer object from a first remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured or runs on the

first remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

[1.2] at the base computer, receiving data about the computer object from a second remote computer on which the computer object or similar computer objects are stored, wherein said data includes information about events initiated or involving the computer object when the computer object is created, configured, or runs on the second remote computer, said information including at least an identity of an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed;

[1.3] storing, at the base computer, said data received from the first and second remote computers;

[1.4] correlating, by the base computer, at least a portion of the data about the computer object received from the first remote computer to at least a portion of the data about the computer object received from the second remote computer;

[1.5] comparing, by the base computer, the correlated data about the

computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities; and [1.6] classifying, by the base computer, the computer object as malware on the basis of said comparison.

(*Id.* 20:27-21:3.)

43. An example embodiment of the claimed method may work as follows. A base computer receives event data from a first and second computer about an Object X. The first computer reports that X was created by Object PARENT, and the second computer reports that X created Object CHILD. The base computer correlates the data from the first and second computer as being about object X. The base computer then compares CHILD to another object in the community (Object MAL) to identify their relationship. For example, the base computer determines that CHILD is MAL's parent, in part by comparing the keys for CHILD and MAL to determine that the two share a "gene." Based on that comparison, the base computer classifies X (and CHILD AND PARENT) as malware.

44. The dependent claims build upon the independent claims.

- Dependent claims 2-7 (and corresponding claims 16-21) concern the identified relationships between the objects and how those relationships are used.

- Claim 2 recites that the identified “relationship” (*e.g.*, between CHILD and MAL) is either “direct” or indirect (*i.e.*, “via other objects”).
- Dependent claim 3 recites that, based on the identified relationship, the based computer monitors the related objects (*e.g.*, CHILD and MAL) “as a homogeneous entity” and identifies one as malware.
- Dependent claim 4 recites that the base computer deduces that an object is malware based on the behavior of a related object (*e.g.*, deduce that X, CHILD, or PARENT is malware based on MAL being malware).
- Dependent claim 5 recites classifying the object (*e.g.*, X) as malware if it is related to at least one other object classified as malware (*e.g.*, CHILD).
- Dependent claim 6 recites that the other object (*e.g.*, MAL) is an object or similar object stored on the first or second computers.
- Dependent claim 7 recites that the other object (*e.g.*, MAL) can be the parent, child, or other relation (*e.g.*, twin) of the object (*e.g.*, X).
- Dependents claims 9-13 and 23-27 detail the “key” used to identify an object.
 - Claim 9 recites that the data must come in the form of a “key”

obtained by a “hashing process.”

- Claim 10 recites that the key includes at least “one component representing executable instructions,”
- Claims 11-12 recite that the key includes at least “one component that represents data about said computer object” including various metadata items.
- Claim 13 recites that the key includes at least “one component that represents the physical size of the computer object.”
- Dependent claims 8 and 22 recite that the information about the object (*e.g.*, X) sent from the remote computers must include certain enumerated data.
- Dependent claims 14 and 28 recite generating a “mask” defining “acceptable behavior” for an item classified as *not* malware, monitoring the object’s operation, and if the monitored behavior extends beyond the behavior permitted by the mask, reclassifying the item as malware.
- Dependent claim 29 recites a “computer program product” that performs the method of claim 1.
- Dependent claim 30 recites that the data received about the computer object from the first remote computer differs from the data received about the computer object from the second remote computer.

B. Claim Construction

45. Petitioner states that it relies on the plain and ordinary meaning of the claims. (Petition 6.) In the parallel district court litigation, Petitioner and its co-defendants argued for constructions of various terms, but the district court construed each to have its plain and ordinary meaning, and made the following clarifying remarks:

Terms	Construction
“remote computer” “first remote computer” “second remote computer” “similar objects” “similar computer objects”	Plain-and-ordinary meaning (EX-2010 3, 5, 8.)
“events initiated [by] or involving the computer object”	Plain-and-ordinary meaning Note: the plain-and-ordinary meaning of “events” includes but is not limited to “actions or behaviours of an object acting upon another object or some other entity” (<i>Id.</i> 5.)
“correlating...at least a portion of the data about the computer object...comparing...the correlated data about the computer object”	Plain-and-ordinary meaning. Not <i>all</i> of the data must be correlated before comparing. (<i>Id.</i> 6-7.)
“a mask for said computer object that defines acceptable behaviour for the computer object”	Plain-and-ordinary meaning. The claim language recites that at least a portion of the data, but not all of the data, must be correlated before comparing. (<i>Id.</i> 6-7.)

C. Level Of Ordinary Skill In The Art

46. I understand that certain issues relating to the validity of the '389 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.

47. In my opinion, a POSITA at the time of the invention of the '389 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, or Computer Science or in a similar discipline, and have 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 and/or training in a related discipline can compensate for shorter work experience. (*Id.*)

48. 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 (PHOSITA) 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, 6-7.)

49. Despite minor variations in these two positions, these variations would not alter my opinions expressed in this Declaration.

50. Based on either standard, I was at least a person of ordinary skill in the art at the time of the invention of the '389 patent.

V. OVERVIEW OF PRIOR ART

A. Overview Of Kester

51. Kester teaches that its “invention is related to computing devices and, more particularly to monitoring and controlling application files operating thereon.” (EX-1004 [0003].) Kester explains that the increase in Internet connectivity poses

risks to employers, because employees are able to download and install illegal or unlicensed applications themselves, run processes that monopolize network bandwidth, and/or expose the employer to risk to sensitive data. (EX-1004 [0005].) For example, instant messaging (IM) can increase productivity by facilitating file transfers and communication, but can also pose a risk to sensitive data. (*Id.*) As another example, “streaming media is a growing concern because of its drain on network bandwidth.” (*Id.*)

52. While software was available to manage how employees access the Internet in the workplace, Kester sought to provide a solution capable of managing “the broader intersection of employees with their computing environments.” (EX-1004 [0006].) Accordingly, Kester discloses a system and method for controlling both the launch of applications and network access requests. (*Id.* [0040]-[0041].)

53. Consistent with Kester’s goal of controlling the “broader” intersection of employees with their computing environments, Kester teaches a method that is capable of controlling a wide variety of application categories. (EX-1004 Fig. 4A (listing 44 exemplary categories).) Figure 4A lists at least 44 exemplary categories of applications:

FIG. 4A

Parent group	Category	Security VC	Liability VC	Productivity VC
System				
	Operating Systems			
	Device Drivers			
	File Management			
	Installers			
	Miscellaneous Utilities			
Access/Privacy				
	Anti-virus Software			
	Authentication			
	Encryption			
	Firewalls			
	Hacking	X	X	
	Remote Access			
	Spyware	X		
	System Audit			
Productivity				
	Contact Managers			
	CRM			
	Data Warehouse, Analytics, Reporting			
	Database			
	Document Viewers			

	ERP/SCM			
	Graphics			
	Infrastructure			
	Presentation			
	Project Managers			
	Proprietary			
	Reference			
	Search/Retrieval/Knowledge Management			
	Software Development			
	Spreadsheets			
	Suite/integrated			
	Web/Desktop Publishing			
	Word Processing			
Communication				
	Collaboration			
	Dedicated Browsers			X
	Email			
	Instant Messaging			
	P2P Filesharing	X		
	Telephone/Conference/Fax			
	Web Browser			
Audio/Video				
	Media Players			X
	Imaging			X
Entertainment				
	Adult	X		
	Gambling	X		
	Games			X
Malware				
	Malicious Applets and Scripts			

(EX-1004 Fig. 4A, [0081]-[0082].) Consistent with Kester's acknowledgement of risks associated with communication-related applications, Kester discloses communication-related categories, such as "Instant Messaging," "Email," "P2P Filesharing" and others. (*Id.* Fig. 4A, [0005].) Consistent with Kester's acknowledgment of bandwidth problems with streaming applications, Kester discloses audio/video-related categories, such as "media players" and "imaging." (*Id.*)

54. Both Petitioner and Dr. Lee characterize Kester as a method for classifying a computer object as malware based on the computer object's behavior. (Petition 7-8; EX-1003 ¶56.) But it is important to understand that Kester is not directed to the ability to determine whether a process was safe or malware. Rather, as I have explained above, Kester's goal was to provide employers with control across a large variety of applications. (EX-1004 [0006], Fig. 4A.)

55. Malware is merely one of the seven exemplary parent groups of categories, and contains a single exemplary category of "malicious applets and scripts." (EX-1004 Fig. 4A.) This is the sole mention of malware in Kester, which contains no further description of what should be categorized as malicious and/or what policy should be implemented for this category. (EX-1004 Fig. 4A, [0082].) The majority of categories depicted in Kester's embodiment are dedicated to monitoring productivity, communication, or privacy. (*Id.*)

56. Kester's method is focused on controlling when a user or workstation is permitted to launch an application and when and how the application can access a network. (EX-1004 Abstract, [0040]-[0042].) Kester discloses a "workstation management module" ("WMM") that can detect the launch of an application on a workstation and/or a network access request and determine an access privilege. (EX-1004 [0040]-[0041].) Recognizing that risk may be particular to the individual user or particular workstation, Kester teaches that the access privilege may be unique to

the workstation and/or user. (EX-1004 [0042], [0057] (“module 208 applies the policy(s)/rule(s) associated with the hash that matches the application and/or the user requesting the application or network access”), [0059] (“module 208 can stop execution or network access for the application depending on a policy associated with the user at this workstation”).)

57. If “the application is allowed to run on the workstation,” the “workstation management module” then “monitor[s] the ongoing network activity or behavior of the application.” (*Id.* [0041].) “For example, each time an application ... attempts to access a network, the workstation management module determines whether to allow [the action].” (*Id.*) Again, access privileges can be associated with the application directly or they can be applied based on the application’s category. (*Id.* [0048].)

58. For example, a user (and/or their workstation) in the finance department may have access to an application that can access and adjust payroll, whereas another user in a different department may be subject to a different policy regarding that or a similar application. *See, e.g.*, EX-1004 [0057] (“module 208 applies the policy(s)/rule(s) associated with the hash that matches the application and/or the user requesting the application or network access”), [0059] (“module 208 can stop execution or network access for the application depending on a policy associated with the user at this workstation”).

59. Figure 1 of Kester illustrates an example of multiple workstations (1 through N) linked together on a local area network 100, with each workstation equipped with its own WMM:

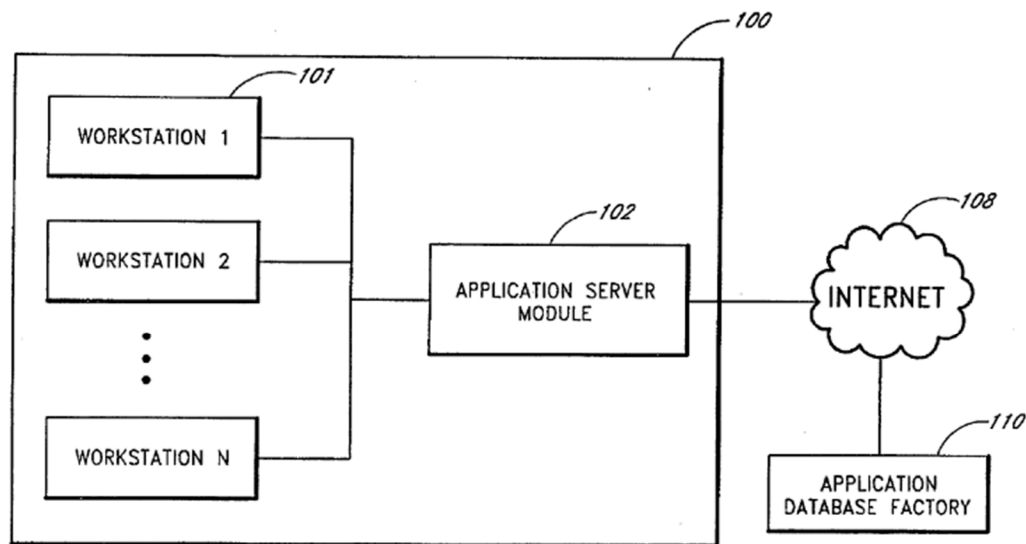


FIG. 1

(EX-1004 Fig. 1.) Kester teaches that each WMM may include a hash/policy table that allows determination of an appropriate policy for each application and/or request. (EX-1004 [0039].)

60. The hash/policy table can include a list of application names, publishers, suites, hashes, ports, protocols, I.P. addresses, categories, and rules or policies associated therewith. (EX-1004 [0054].) For example, in one embodiment, the table comprises a list of hash values associated with corresponding policies, as well as categories that are associated with the hash values and/or policies. (*Id.*) In another embodiment, the table includes names/publishers/suites or other properties which

identify the application instead of hash values. (*Id.*) And in another embodiment, the table includes the port/I.P. address/protocol or other properties which identify the applications in the hash/policy table. (*Id.*) Kester's ultimate goal is that, when a user launches an application or when the application or user requests access to the network, the policy which corresponds to that application or user can be easily identified to determine what access privileges apply to the request. (EX-1004 [0055].)

61. For example, when the WMM detects the launch of an application or a request to access a network, a "hash" or "application digest" is created, based on unique properties related to the application or user requesting access. (EX-1004 [0140]-[0142], Fig. 14 (step 1404), [0051].) The WMM then searches the hash/policy table for a corresponding hash that matches the network request. (EX-1004 [0053], [0142], Fig. 14 (step 1406).) If a match is found, then the policy associated with the hash is applied to the application in response to the network access request. (EX-1004 [0143], Fig. 14 (step 1410).)

62. However, prior to the WMM consulting the hash/policy table, Kester requires that a human reviewer make certain "predetermined associations" between the application/request and its expected network activity, any applicable categories, and a policy. (EX-1004 [0040] ("predetermined association between the launched application and one or more categories"), [0043] ("predetermined association between the application and an expected network behavior or activity"), [0072] ("The

network administrator can...select or create access privileges/policies/rules for users, workstation, and/or groups of users/workstations.”).)

63. Using the monitored “network activity,” the workstations and server-side components can learn each application’s “*expected* network behavior.” (*Id.* [0043]-[0045].) The “expected network behavior” can be based on the prior access patterns of the application (or of a related application, such as an earlier version) on the workstation or across multiple workstations. (*Id.* [0043]-[0045], [0069]-[0070].) The workstations capture and log those prior access patterns in a “logging database” and report that data to the server-side components. (*Id.* [0045], [0051], [0062], [0069]-[0071].)

64. Kester teaches that determining the expected network activity may involve “merging and sorting” the data about the application received from the different workstations. (*Id.* [0074]. [0087].) Kester provides only two examples of “merging and sorting”: (1) merging and sorting network access data based on “destination IP address” (*id.* [0162]); and (2) merging and sorting application access data based on an application’s “request frequency” (*id.* [0119]).

65. Kester discloses an “application analyst’s classification module” with which the human reviewer may interface to determine these associations. (EX-1004 [0178].) Figure 20 of Kester illustrates “a process for analyzing the network access data associated with an application at the application database factor.” (EX-1004

[0178].) In particular, Kester describes two different determinations. (*See generally* EX-1004 [0178]-[0187].)

66. For one of these determinations, after the reviewer is provided with a list of applications from the “uncategorized application database” and/or “network access database,” the reviewer determines the appropriate category or categories of the application. (EX-1004 [0178], [0134].) Figure 13 further elaborates on this specific determination of categorizing an application. (EX-1004 [0134], Fig. 13.) The application analyst classification module displays the application and any related data on the graphical user interface (GUI) for the human to review. (EX-1004 [0134].) “[T]he human reviewer uses the application, related information, and any Internet information to research the application.” (EX-1004 [0135].) “The human reviewer can further review documents, specifications, manuals, and the like to best determine the category or categories to associate with the application.” (*Id.*) The reviewer then categorizes (or classifies) each application using this evidence, “any hints from the related information,” and/or other research. (*Id.*)

67. As I explained above, this categorization determination could include any one or more of the 44 exemplary categories disclosed by Kester related to “productivity,” “audio/video,” “communication,” etc. (EX-1004 Fig. 4A.) While not included in Figure 4A, Kester further discloses two additional categories: “expected or predetermined network access” and “unexpected network access.” (EX-

1004 [0178].) Thus, Kester may use the term “expected” or “predetermined” when referring to the general category of expected network access. (*See id.*)

68. Returning to Figure 20, for the other determination, Kester teaches that the human reviewer again reviews the application and any related information on the application analyst’s classification module’s GUI “to determine an expected or allowed network activity.” (EX-1004 [0179]-[0182], Fig. 20 (steps 2004, 2006).) The network activity of the same application running on different applications can be weighted in a predetermined manner. (EX-1004 [0183].) Kester explains that the “expected network activity for a given application can include one or more network attributes,” such as a specific protocol, a specific I.P. address, and a specific access port. (EX-1004 [0184].) Once the human reviewer has determined this expected network activity, the WMM stores the expected network activity in the hash/policy table. (EX-1004 [0183].) Kester teaches that for the same application, network attributes may be different for each attempted access. (EX-1004 [0185].) That is, Kester also uses “expected” or “predetermined” to refer to the expected network activity/attributes that are unique to each entry of the hash/policy table. (*See* EX-1004 [0185].)

69. Thus, the network administrator determines a policy specific to each combination of network attributes and “[e]ach combination of the one or more attributes can be associated with one or more categories.” (EX-1004 [0185]-[0186].)

And as I discussed above, this policy or category determination may be specific to the user or workstation running the application and/or making the network request. (EX-1004 [0042], [0186].)

70. Kester provides an illustrative example to help put these two different uses of “expected” within context:

For example, a first entry in the application inventory database 103 corresponds to the application when the application is operating in a predetermined manner. These expected features relate to normal network activity by the application. A second entry corresponds to the same application hash but different parsed properties or features associated with the application when the application is not operating in a predetermined manner.

(EX-1004 [0070].)

71. Thus, a POSITA would understand that Kester first uses “expected” or “predetermined” to refer to the expected network activity (or “expected features”) that are unique to each entry in the hash/policy table. (*See id.* [0070], [0179]-[0184].) That is, if an application is operating in an expected or predetermined way, then there will be an entry in the hash/policy table which matches the network attributes associated with that operation. (*See id.*) But Kester also uses “expected” or “predetermined” to refer to the “expected network access” category—*i.e.*, in the example of paragraph [0070], whether the features for a particular application

operation are “operating in a predetermined manner” and “relate to normal network activity by the application.” (*See* EX-1004 [0070], [0178], [0134]-[0135].)

72. As another example, you could have a hypothetical application request, which is an “expected network activity”—*i.e.*, the network attributes associated with that request match the “expected” or “predetermined” network attributes that are unique to an entry in the hash/policy table. (*See* EX-1004 [0070].) This same request could also be categorized as “unexpected network access”—*i.e.*, that network attribute combination, while known and recognized, was determined by the human reviewer to be “unexpected network access” for that application. (*See id.* [0070]. [0178].) Accordingly, you would have an application or request that is both “expected network activity” and “unexpected network access.”

73. If a server-side component (e.g., “application server module”) determines that “[a] categorized application ... does not operate in a predetermine manner,” it may place the application in an “uncategorized application database.” (*Id.* [0070].) The human network administrator then reviews data about the uncategorized application to categorize it. (*Id.* [0071]-[0072], [0134]-[0136].)

74. While one might question the need for categories on top of the initial determination as to whether an application matches an entry in the hash/policy table, a POSITA would understand that the variety of categories was critical to Kester’s stated goal—providing employers with a system and method for controlling

applications files across the “broader intersection” of employees with their computing environment. (EX-1004 [0006].) By allowing a human reviewer to perform an in-depth review of each application and request, within the context of the particular user(s), workstation(s), and employer’s policies, the reviewer could make an informed decision of any and all applicable categories across all types of applications/requests. (EX-1004 [0006], Fig. 4A, [0081]-[0082].) That is, Kester could provide the “broader” “monitoring and controlling” invention it claimed. (EX-1004 [0003], [0006].)

B. Overview Of Kennedy

75. Kennedy discloses a method of “detecting malicious software through file group behavior.” (EX-1005 Title.) Rather than a community-based approach (where threat data is aggregated from many computers on a network), Kennedy utilizes a local approach where a single computer uses a “malicious software detection module (MSDM)” to identify suspicious files, monitor their behavior, and ultimately categorize them as malicious. (*Id.* 1:42-43.)

76. Kennedy discloses a “malicious software detection module” (or “MSDM”) that operates on a computer connected to a network. (EX-1005 1:42-52) The MSDM runs on a computer system and “monitors a storage device of the computer system for the arrival of software from a suspicious portal.” (EX-1005 1:45-48.) Kennedy’s MSDM is implemented through five modules, as illustrated below:

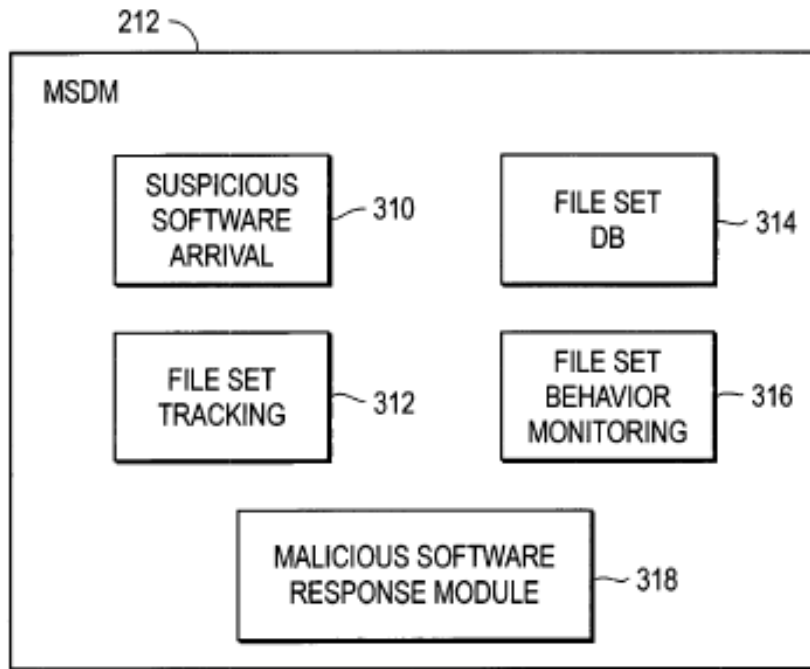


FIG. 3

(EX-1005 Fig. 3.)

77. The “arrival module” “monitors the storage device [of the computer] and detects the arrival of new software ... via one or more entry points, or ‘portals.’” (*Id.* 3:47-59.) If software arrives by a “suspicious” portal (e.g., an “open drive share portal”), the MSDM marks each file of that software as “suspicious.” (*Id.* 4:5-18.)

78. The “file set tracking module” tracks the “suspicious software” to identify “a set of files” associated with that software. (*Id.* 4:19-20.) Initially, “[t]he set [of files] includes each file that constitutes the suspicious software.” (*Id.* 4:22-23.) The “file set tracking module” then “tracks the behaviors of the files in the [suspicious] set and records each time a file in the set ‘touches’ another file on the storage

device or elsewhere in the computer system.” (*Id.* 4:23-26.) To “touch” can “mean[] to create, modify, and/or read a file,” or to “fewer and/or more actions,” such as “creating and modifying an executable file and/or a file over a specified size.” (*Id.* 4:26-34.) Each time a file is touched, “[t]he file set tracking module adds the touched file to the set of the file that touched it.” (*Id.* 4:33-34.) Thus, the system “creates a logical set of files associated with the suspicious software,” where “[t]he set contains files that form the suspicious software and any file that is touched by a file already in the set.” (*Id.* 5:57-63)

79. The “file set behavior monitoring module 316 monitors the behavior of the files within the set associated with each suspicious software” both individually and collectively. (EX-1005 4:64 – 5:3.) To determine whether software represented by the file set is malicious, “[t]he file set behavior monitoring module 316 employs one or more heuristics” that describe “one or more conditions that, if satisfied, indicate that the software is malicious.” (EX-1005 5:4 – 11.) If the heuristics determine that an object is malware, the malicious software response module 318 performs one or more remedial actions, such as alerting the end user of the client computer system, quarantining the software, preventing replication, or performing secondary tests on the software. (EX-1005 5:12 – 42.)

80. The MSDM’s malicious software detection steps are illustrated in Figure 4:

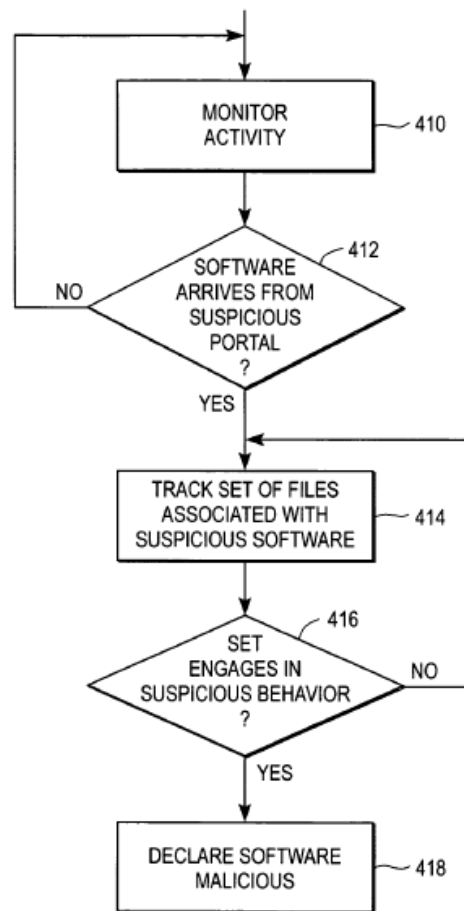


FIG. 4

(EX-1005 Fig. 4.)

81. “The MSDM 212 monitors 410 activity on the local storage device 108.” (EX-1005 5:51-52.) The computer system’s MSDM will initially classify an object received by that computer system as suspicious if “the software arrived through a portal designated as suspicious.” (EX-1005 5:52–56.) Next, the MSDM tracks files on the local storage device that are associated with the software it has designated as suspicious and, if the files in the set either individually or collectively engage in malicious behavior, the MSDM classifies “the software [as] malicious and

notifies the end-user, quarantines the software, blocks the software from performing certain activities, and/or performs other actions.” (EX-1005 5:57–6:4.)

C. Overview Of Honig

82. Honig discloses a computer system architecture that supports the automation of certain steps in deploying “data-mining” intrusion detection systems (IDSs). (EX-1005 4:53-56.) Data-mining IDSs collect data from sensors monitoring a system and use machine learning algorithms on a large set of “training data” to build detection models. (EX-1005 2:5-16.)

83. The type of training data needed depends on the type of detection model (and, therefore, the type of machine learning algorithm) that used to detect intrusions. Honig discloses at least three general types of model generation algorithms: (1) “misuse detection,” which models known attack behavior and trains on labeled normal and attack data; (2) “supervised (traditional) anomaly detection,” which detects activity that diverges from a model of normal activity and trains on normal data; and (3) “unsupervised anomaly detection,” which also detects activity that diverges from a model of normal activity but which trains on unlabeled data. (EX-1005 2:29-49, 20:33-38.)

84. Honig’s architecture consists of: sensors and detectors for collecting information from the monitored system; a data warehouse which stores the collected data; model generators which access the data and train data mining-based detection

models; model distributors which transfer the models to detectors; and analysis engines to provide data analysis capabilities. (EX-1005 6:64-7:7, 1:44-47, 4:53-56, 3:24-30.)

VI. KESTER IN VIEW OF KENNEDY AND HONIG DOES NOT DISCLOSE OR RENDER OBVIOUS THE CHALLENGED CLAIMS

85. Petitioner and its expert allege that Kester in view of Kennedy renders obvious every limitation of independent claims 1 and 15, and this allegations underlies their challenge of all remaining dependent claims 2-14 and 16-30.

Ground 1	Section 103(a)	Obvious over Kester in view of Kennedy	Claims 1–13, 15–27, 29, 30
Ground 2	Section 103(a)	Obvious over Kester in view of Kennedy and Kennedy	Claims 14 and 28

86. As I discussed above, I apply the plain and ordinary meaning of all claim terms, consistent with the district court’s claim construction order. (*See* Section IV(B), *supra*.)

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

VII. GROUND 1: KESTER IN VIEW OF KENNEDY DID NOT RENDER CLAIMS 1-13, 15-27, AND 29-30 OBVIOUS.

88. In Ground 1, Petitioner proposes a combination of Kester and Kennedy whereby (1) Kester’s workstations use Kennedy’s “file set tracking module” to

identify “file sets” of related files, (2) those workstations then send data about those file sets and their behaviors to Kester’s “application server module,” and (3) Kester’s “application server module” uses Kennedy’s “behavior monitoring module” to apply Kennedy’s classification heuristics to the received data, thus automatically classifying certain files as malware. (Petition 28-29.)

89. Petitioner’s combination suffers from two shortcomings. *First*, neither of the references (alone or in combination) teaches or suggests the claimed idea of identifying relationships between objects based, at least in part, *on a comparison of the objects* (e.g., detecting common “genes” in parent and child objects). *Second*, Petitioner discusses each claim limitation with respect to only one reference in isolation, failing to explain how (or why) *the combination* satisfies every limitation, including the “receiving,” “correlating,” and “comparing” steps. The Board may deny the petition for either reason (or both).

A. Neither Kester nor Kennedy teaches that identifying a relationship between two objects includes comparing those objects, as required by the independent claims.

90. One shortcoming of Petitioner’s combination is that neither Kester nor Kennedy—alone or in combination—teaches or suggests the idea of “identifying relationships” between two objects by “comparing” those objects, rather than by merely comparing the origins or activities of those objects.

91. The '389 patent discloses that the invention “tracks which objects are related to each other and uses the concept of ancestry to enable objects to be marked as malware.” (EX1001 17:22-24.) It further discloses that whether two objects are “related” may depend not only on event metadata (*e.g.*, that a “process may spawn child processes which are therefore related”) but also on object data, such as shared “genes” (*i.e.*, “a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object,” “referred to herein as a ‘gene’”). (*Id.* 17:26-30.)

92. The challenged claims recite this idea of determining relationships between objects by *comparing* data about those objects rather than by simply detecting events related to a single object. Specifically, they require “comparing ... the correlated data about the [first] computer object ... to other objects or entities to identify relationships between the correlated data and the other objects or entities.” (*Id.* 20:56-60.) This recited comparison “to other objects or entities” invokes procedures described in the specification, such as comparing two objects’ respective object data to determine whether they share a given “gene” and are therefore related. (*Id.* 17:22-31.)

93. Neither Kester nor Kennedy teaches or suggests this limitation. Petitioner’s position is unclear on this issue because, at different points, Petitioner seems to argue that the “comparing” limitation is taught either by (1) Kester’s comparison

of an application's behavior to that *same application's* expected behavior (Petition 26) or instead, (2) Kennedy's determination that an object is related to known malware (*id.* 28-29.) Regardless, neither of these teachings discloses the claimed comparing step.

1. Kester does not teach the claimed “comparing.”

94. Petitioner's first argument is that Kester teaches the “comparing” limitation by comparing an application's behavior to the *same application's* expected behavior. (Petition 26.) This argument is wrong for at least two reasons.

95. *First*, rather than a comparison “to other objects,” Kester's comparison is (at best) a comparison of data about *the same* object. Petitioner's argument is that Kester compares an application's behavior to the *same application's* expected behavior. This plainly does not meet the claimed limitation of comparing data about a computer object “to *other* objects or entities.” (*Id.* 20:56-60.)

96. *Second*, Petitioner does not argue that Kester's comparison of an application's behavior to that same application's expected behavior is made “to identify relationships” between those two objects, as claimed. No such argument is possible because the comparing Petitioner relies on is of data about two objects with an already-known relationship: *they are the same application*. Thus, the “comparing” on which Petitioner relies in Kester is not done “to identify relationships” between objects.

97. Petitioner argues briefly that Kester’s comparing *is* between different objects because an application’s expected behavior can be derived (in part) from the behavior of a different but related application, such as a previous version of the application. (Petition 26.) But this argument does not remedy either of the two shortcomings identified above.

98. As to the first shortcoming (*i.e.*, that Kester compares data only about the *same application*), even if Kester determines that application’s expected behavior, at least in part, based on a related application’s behavior, the resulting expected behavior is still data about *the application*, not about the related application. Thus, comparing that data (about the application) to other data about that same application is not comparing data about the application “*to other* objects or entities,” as claimed.

99. As to the second shortcoming (*i.e.*, that Kester’s comparison of an applications behavior to that same application’s expected behavior is not made “to identify relationships” to other objects or entities), Petitioner’s argument is inapplicable. Indeed, Kester’s comparison involves data about only one application. And it is made to determine whether that application is behaving in an expected manner, not to determine whether the application is *related to* any other application.

100. Thus, Petitioner makes no plausible argument that Kester teaches the claimed idea of “comparing” information about two different objects to identify relationships between those objects.

2. Kennedy does not teach the claimed “comparing.”

101. Unlike Kester, Kennedy does teach the idea of a malware classifier that identifies and considers relationships between different objects (which are used to create the disclosed “file sets”). (EX1005 4:19-63.) But Kennedy identifies those relationships based only on events—*i.e.*, detecting that one file “touched” another—not based on a comparison of information about two different objects, as claimed. (*Id.*) Thus, Kennedy does not disclose the claimed “identify[ing] relationships” to “other objects or entities” by “comparing” data about a first object “to other objects or entities.” (*Id.* 20:56-60.)

102. The petition fails to plainly state what aspect of Kennedy (or of the claimed combination) Petitioner contends discloses the “comparing” limitation. (Petition 27-31.) At one point, Petitioner appears to argue that Kennedy performs the claimed “comparing” by determining that a file in a file set is malware:

103. ... when the *comparison* of [Kennedy’s] collection data to other objects indicates the application is related to known malware, Kennedy’s automated heuristics can classify the application appropriately as malware.

104. (Petition 29.)

105. At his deposition, Petitioner’s expert (Dr. Wenke Lee) confirmed that this is indeed Petitioner’s theory. Asked repeatedly how the combination satisfies the claimed “comparing” limitation, Dr. Lee testified each time that the combination

performs the comparing step by comparing unspecified data about the object being evaluated against unspecified data about other objects that the system has already determined are malware, and, if the data are sufficiently similar, classifying the object as malware. [Wenke Depo. 160:16-162:12; 174:2-175:19; 188:9-190:20].

106. Petitioner’s combination fails to disclose or teach the claimed “comparing ... to identify relationships” step because this purported comparing on which Petitioner’s argument relies is neither (1) taught or suggested by Kennedy nor (2) made “to identify relationships” between the object and other objects.

107. *First*, Kennedy does not teach or suggest, as Petitioner argues, the idea of comparing data about the object being evaluated against data about other objects that the system has already determined are malware. Rather, to determine whether a file is malware, Kennedy applies behavioral heuristics that compare an object’s behavior to *already-known behavioral patterns of malware*. Kennedy discloses the following behavioral heuristics for determining whether software is malware: the software “attempts to copy itself to another computer system via a technique such as email and/or an open drive share,” “sends more than a certain number of emails (e.g., N=10) within a certain time period,” “mines files on its local computer system for email addresses, performs certain network activities such as enumerating for open file shares and/or copying files, makes registry changes related to the persistency of the Software, *etc.*” (EX 1005 5:4-33) None of those heuristics involve “comparing”

the software's behavior to other software that the system previously determined was malware. Thus, Kennedy does not teach or suggest "comparing ... data about the computer object ... to other objects or entities," as claimed.

108. *Second*, even if Petitioner were correct that Kennedy determines that an object is malware by comparing data about that object to data about known malware, its argument would still fail because this purported "comparing" is not done "to identify relationships" between the object and the malware. Instead, the proposed comparing is done to determine that the two objects behave similarly.

109. That two objects are similar is not a "relationship" between the two objects within the meaning of the claims. The specification discloses that "an object is considered to have a relationship *based on the event performed by it upon another object, or upon it by another object.*" (*Id.* 11:11-13). The specification further illustrates the meaning of "relationship" through examples, such as "which object created this object, which objects this object created, which objects created a registry key to run or configure this object, which objects were configured by or had registry keys created by this object, etc." (*Id.* 11:7-11.) Thus, even if Petitioner were correct that Kennedy compares two objects to determine that they behave similarly, that comparison is not made to "identify relationships" between the two objects, within the meaning of the claims.

110. Thus, Petitioner’s combination fails to show that Kennedy (alone or in combination with Kester) teaches the claimed “comparing” of objects and entities to identify relationships between them.

111. Because neither Kester nor Kennedy discloses “identifying relationships” between two objects based on “comparing” information about those objects, Petitioner fails to show that the combination of those references teaches or suggests the claimed “comparing” limitation.

112. Petitioner provides no argument that the missing “comparing” limitation would have been rendered obvious by ordinary skill. Nor does it include any explanation of why a POSITA would have been motivated to introduce that limitation. Thus, Petitioner has not shown that the combination of Kester and Kennedy renders the independent claims obvious.

B. Petitioner fails to show that the proposed combination would meet the claimed “receiving,” “correlating,” and “comparing” steps, as required by the independent claims.

113. A second independent reason to find the challenged claims patentable is that Petitioner does not sufficiently explain the proposed combination of Kester and Kennedy, including how that combination supposedly satisfies every limitation of the independent claims. Petitioner tries to identify each limitation in one reference or the other, but when it comes to combining those references, Petitioner leaves the

Board (and Patent Owner) guessing about how the purported combination might work and why.

114. At a high level, Petitioner argues that, in its combination, (1) Kester’s workstations use Kennedy’s “file set tracking module” to identify “file sets” (2) the workstations then send certain (unspecified) data about the file sets to Kester’s “application server module,” which (3) applies Kennedy’s behavioral heuristics to that data *across multiple workstations* (for some unknown reason and in some unknown way), and thus automatically classifies certain files as malware (rather than by using Kester’s preexisting manual classification procedure). (Petition 28-29.)

115. Petitioner’s description of its combination has many significant gaps. It is unclear how the proposed combination would work, why a POSITA would have been motivated to make it, or how it would satisfy all claimed limitations, including the claimed (1) receiving, (2) correlating, and (3) comparing steps.

1. Petitioner’s combination does not teach or suggest “receiving” the claimed “information about events” from remote computers, as required by independent claims 1 and 15.

116. Petitioner argues that it would have been obvious for Kester’s “application server module” to receive data from Kennedy’s MSDMs, executing on multiple workstations. (Petition 29.) But Petitioner does not say *what data* each MSDM supposedly captures and sends to the “application server module.” This omission creates a threshold problem: Petitioner fails to show that it would have been obvious

for Kennedy’s MSDM to capture and send the claimed “information about events,” including “an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed,” as required by independent claims 1 and 15.

117. Petitioner argues that *Kester* (alone) teaches that the “application server module” receives “information about events”—specifically, information indicating that a given application (actor object) has attempted to access (event type) a particular network address (target “entity”). (Petition 13-19.) *But this is not the data Petitioner argues would be “correlated” and “compared” to “classify” an object as malware.* (Petition 27.) Instead, Petitioner asserts that those steps would be performed on data sent *from Kennedy’s MSDM*, as it operates on Kester’s workstations. Thus, for Petitioner combination to render the claims obvious, Petitioner must show that Kennedy’s MSDM captures the claimed information about events or that it would have been obvious to modify the MSDM to do so and to send the information to Kester’s server-side components. But Petitioner does neither.

118. For instance, Petitioner asserts that each workstation would include Kennedy’s “file set tracking module” (Petition 28), but that module merely identifies “file sets,” not the “information about events” recited in the claims. Thus, adding Kennedy’s “file set tracking module” to Kester’s workstations would not enable

those workstations to capture (let alone send) the claimed “information about events.”

119. Even if Kennedy’s “file set tracking module” does record (at least temporarily) “each time a file in the set ‘touches’ another file” (EX1005 4:19-36.), Kennedy does not teach that the “touch” data captured by the file set tracking module includes the claimed “information about events” (*e.g.*, includes “event type”). Nor does Kennedy teach that this “touch” data is ultimately used by Kennedy’s classification heuristics. To the contrary, Kennedy teaches that those heuristics classifying files based on data captured by a *different* module—*i.e.*, the “file set behavior monitoring module”—which Petitioner has proposed would be placed on Kester’s *servers*, not on Kester’s workstations. (*Id.* 4:64-5:3.) In short, the information captured by the “file set tracking module” does not disclose the claimed “information about events.”

120. Thus, in the proposed combination, there is no teaching or suggestion that Kester’s “application server module” would receive the recited “information about events” from Kennedy’s MSDM.

2. The proposed combination does not teach or suggest “correlating” event data from Kennedy’s MSDM.

121. The proposed combination is also missing critical explanations about how or why Kester’s “application server module” would “correlate” the data it receives from Kennedy’s MSDMs.

122. Petitioner argues that Kester (alone) teaches correlating “information about events” because Kester “merges and sorts” network access events based on destination IP address. (EX1004 [0162].) But as before, Kester’s is not the data that Petitioner has proposed would be “correlated” and “compared” with other objects to “classify” an object as malware. (Petition 27.) Instead, Petitioner proposes these steps would be performed on the data *sent from Kennedy’s MSDM*. But Petitioner does not explain *how* or why Kester’s “merging and sorting” would be applied to that data.

123. A threshold problem (as shown in the previous section) is that Petitioner is unclear about what data (in the proposed combination) is sent from Kennedy’s MSDM. Nevertheless, for purposes of analyzing this limitation, Patent Owner assumes in this section (without conceding) that the data sent includes everything that Kennedy’s “file set behavior monitoring module” (executing on Kester’s server-side components) would need to perform the classification heuristics disclosed in Kennedy. (EX1005 5:4-33.) That data includes at least the “file set” definitions and what Petitioner calls the “related file behavior.” (Petition 29.)

124. But, even with those assumptions, the petition leaves unexplained *why* and *how* a skilled artisan would apply Kester’s “merging and sorting” techniques to Kennedy’s data. Consider Kennedy’s heuristic that software is malicious if it “sends more than a certain number of emails (e.g., N=10) within a certain time period.”

(EX1005 5:4-34.) The data for that heuristic is presumably the number of emails each application sends over time. How and why would a POSITA apply (to that data) Kester teaching of “merging and sorting” network access events by “destination IP address?” (EX 1004 [0162].) Petitioner does not say.

125. Petitioner’s position is simply that, because Kester applies certain merging and sorting techniques to certain data for a certain purpose, it would have been obvious for the proposed combination to apply *any* merging and sorting techniques to *any* other data for *any* other purpose. This is insufficient to set out a proposed combination of references, let alone to explain why a POSITA would have been motivated to make that combination.

126. In short, Petitioner does not explain how Kester’s “merging and sorting” techniques would be applied to the unspecified data collected from Kennedy’s MSDMs, or why a person of ordinary skill would have been motivated to make that application. Thus, Petitioner has failed to show that the proposed combination of Kester and Kennedy would have rendered it obvious to correlate the claimed event data received from multiple remote computers.

3. The proposed combination does not teach or suggest “comparing ... correlated data ... to other objects or entities to identify relationships between the correlated data and the

other objects or entities,” as required by independent claims 1 and 15.

127. The proposed combination also does not teach or suggest “comparing” the correlated data to another object or entity to identify relationships between the two.

128. Neither Kester nor Kennedy alone teach that limitation. As discussed above (*supra* § V.A.1), Kester’s comparison of one application’s historical behavior to the same application’s current behavior does not satisfy the “comparing” step at least because it is (1) a comparison of data about *the same* object rather than about different objects, and (2) not made “to identify relationships” between different objects. And as further discussed above (*supra* § V.A.2), Petitioner fails to show that Kennedy remedies Kester’s shortcomings because Petitioner’s argument relies on ideas that (1) Kennedy does not teach and (2) fail to suggest any comparison “to identify relationships” between objects or entities.

129. An additional problem surfaces when the *proposed combination* of Kester and Kennedy is considered. Specifically, Petitioner fails to explain what role Kester’s purported “comparing” plays in the proposed combination.

130. Kester’s purported “comparing” (*i.e.*, of an application’s behavior to its expected behavior) is part of Kester’s manual classification technique. Specifically, Kester compares the application’s behavior to the application’s expected behavior to determine whether the application is not operating “in a predetermined manner”

and to consequently place the application in an “uncategorized application database” for later manual classification. (EX1004 [0070].) But the *proposed combination* does not use Kester’s classification technique. Instead, Petitioner proposes that Kester would classify an application *automatically, using Kennedy’s behavioral heuristics* rather than manually, using Kester’s classification approach. (Petition 28-29, 32) (“In this combination, Kester’s application server ... utilizes Kennedy’s automated heuristics.”)

131. But if the proposed combination does not use Kester’s classification technique, then it does not use Kester’s purported “comparing” (which is part of that technique). Thus, the purported “comparing” Petitioner identifies in Kester is irrelevant to whether the *proposed combination* of Kester and Kennedy renders the claims obvious. (Petition 26.)

132. The example Petitioner provides to illustrate the combination (Petition 29) confirms that Kester’s purported “comparing” is not part of the proposed combination. The example posits that, when the server classifies a given executable (“sus-a.exe”) as malware, “*using the file set behavior monitoring of Kennedy*,” a related executable (child “sus-b.exe”) is also marked as malware. (*Id.*) But this is simply how Kennedy alone worked. The only change is that the heuristics/classification occurs on Kester’s server rather than on the workstation. The purported combination thus does not use Kester’s classification technique, including its purported

“comparing.” The “comparing” Petitioner purports to identify in Kester is therefore irrelevant to whether that proposed combination rendered the claims obvious.

133. * * *

134. In sum, the Board should reject Petitioner’s arguments for the second, independent reason that Plaintiff fails to articulate how the proposed combination would work, why a skilled artisan would have been motivated to make that combination, and how the combination would supposedly satisfy every limitation of the independent claims.

C. Petitioner fails to show that the proposed combination would have rendered the dependent claims obvious.

135. Petitioner’s analysis of dependent claims 2-7 (and corresponding claims 16-21) again highlights the deficiencies in how Petitioner has explained the proposed combination of Kester and Kennedy. In short, while Petitioner’s analysis of the independent claims relies on similarity “relationships” purportedly identified by comparing objects to known malware, Petitioner’s analysis of these dependent claims instead relies on Kennedy’s “touch” “relationships,” which are not (and Petitioner never argued are) identified by comparing two objects.

136. Petitioner’s analysis of dependent claims 8 and 22 is similarly inconsistent with Petitioner’s analysis of the independent claims. (Petition 39.) The dependent claims recite that the “comparing ... to identify relationships” recited in the independent claims is done using certain information about the computer object,

such as the object’s “executable instructions.” Petitioner argues that Kester’s workstations collect and send such information to Kester’s server-side components (*id.*), but critically, Petitioner does not even attempt to show that the proposed combination compares that data to other objects or entities to identify relationships, as claimed.

1. Petitioner’s analysis of dependent claims 2-7 and 16-21 is incompatible with its analysis of the independent claims.

137. Petitioner’s analysis of dependent claims 2-7 (and corresponding claims 16-21) again highlights the deficiencies in how Petitioner has explained the proposed combination of Kester and Kennedy. In short, while Petitioner’s analysis of the independent claims relies on similarity “relationships” purportedly identified by comparing objects to known malware, Petitioner’s analysis of these dependent claims instead relies on Kennedy’s “touch” “relationships,” which are not (and Petitioner never argued are) identified by comparing two objects.

138. Dependent claims 2-7 (and corresponding claims 16-21) all concern the identified “relationships” between objects and how those relationships are used in malware classification. The claimed details about the relationships include that they are either “direct” or “via other objects” (claims 2 and 16), that they are parent, child, other process relations (claims 7 and 21), and/or that the related object is an object or similar object stored on the first or second computers (claims 6 and 20). The claimed details about how those “relationships” are used include the ideas of

monitoring related objects “as a homogeneous entity” (claims 3 and 17), deducing that an object is malware based on the behavior of the related object (claims 4 and 18), and classifying the object as malware if the related object is malware (claims 5 and 19).

139. For these limitations, Petitioner relies primarily on Kennedy’s idea of identifying “sets” of related files based on the files “touching” one another, tracking the behavior of those sets as a homogeneous entity, and marking all files in the set as malware when any of the files is discovered to be malware. (Petition 33-39.)

140. The problem with this argument is that Kennedy’s “file set” relationships are not the “relationships” Petitioner relied on in its analysis of the independent claims to purportedly show “comparing ... to identify relationships.” As discussed above (*supra* § V.A.2), Petitioner argued that, in the proposed combination, “comparing ... to identify relationships” (as recited in the independent claims) purportedly happens by comparing an object’s behavior to that of known malware and thus determining that the two behave similarly (and are therefore purportedly “related”).

141. And yet, when the time comes to show that those same “similarity” relationships (purportedly derived from the claimed “comparing ... to identify relationships”) have the characteristics recited in the dependent claims (*e.g.*, are parent/child) or that they are used in the ways recited in those claims (*e.g.*, to trigger

monitoring of the related objects as a “homogenous entity”), Petitioner pivots to arguing that the “relationships” are something different altogether: Kennedy’s “file set” relationships. (Petition 32-39, 43-44.)

142. The similarity “relationships” used in the Petitioner’s analysis of the independent claims cannot satisfy the limitations of dependent claims 2-7 (and 16-21) because they are simply incompatible. Those similarity “relationships” are that an object behaves similarly to known malware. To the extent that is a “relationship” within the meaning of the claims (it is not), the relationship is not one that can be described as “direct” or “via other objects” (claims 2 and 16), a “parent,” “child” or “otherwise process-related” (claims 7 and 21), or to an instance of the object (or similar object) stored on the first or second computers (claims 6 and 20). (EX-1001 3:37-47; 11:5-44; EX-1005 4:19-28.) Moreover, the claimed references do not suggest that, based on these similarity relationships, an object labeled as malware for being similar to known malware would be monitored with the known malware “as a homogeneous entity” (claim 3), be marked as malware based on the subsequent behavior of the malware (claim 4), or be classified as malware if the known malware is malware (claim 5). (*See* EX-1005 4:19-34, 4:64-5:3)

143. Conversely, the Kennedy “file set” relationships Petitioner uses to analyze the dependent claims are not identified by the claimed “comparing.” Indeed, when Petitioner analyzed the independent claims, it never argued that the purported

“comparing” creates Kennedy’s “file set” relationships. It could not because Kennedy’s “file set” relationships are based on sets of files that suspicious software “touched,” not on comparing two files.

144. The Board should therefore reject Petitioner’s analysis. Either the purported combination does not “compar[e] ... to identify relationships” (as recited in the independent claims) or those relationships lack the characteristics and uses recited in the dependent claims. Petitioner cannot have it both ways.

2. Petitioner has not shown that the data used for the “comparing” includes any of the information recited in dependent claims 8 and 22.

145. Petitioner’s analysis of dependent claims 8 and 22 is similarly inconsistent with Petitioner’s analysis of the independent claims. (Petition 39.) The dependent claims recite that the “comparing ... to identify relationships” recited in the independent claims is done using certain information about the computer object, such as the object’s “executable instructions.” Petitioner argues that Kester’s workstations collect and send such information to Kester’s server-side components (*id.*), but critically, Petitioner does not even attempt to show that the proposed combination compares that data to other objects or entities to identify relationships, as claimed.

146. As shown above (*supra* § V.A.1), Kester uses that information only to determine whether an application’s behavior differs from its expected behavior—not to identify any relationship between different objects.

147. As further shown above (*supra* § V.A.2), Petitioner has advanced an alternative theory wherein Kennedy performs the “comparing” by comparing an application’s behavior to known patterns of malware behavior (e.g., emailing a certain number of times per hour), but Petitioner has identified no suggestion in Kennedy that this comparing should use information about objects collected by Kester. If any such argument is available, it was Petitioner’s burden to make it and to provide a rationale for why a skilled artisan would have recognized the combination and been motivated to make it.

VIII. GROUND 2: KESTER IN VIEW OF KENNEDY AND HONIG DID NOT RENDER CLAIMS 14 AND 28 OBVIOUS.

148. In Ground 2, Petitioner argues that a combination of Kester, Kennedy and Honig renders claims 14 and 28. With respect to the claim limitation “generating a mask for said computer object that defines acceptable behavior for the computer object,” Petitioner relies on a combination of Kester and Honig. (Petition 56-60.) However, the combination of Kester and Honig does not render obvious a mask that defines acceptable behavior.

149. Petitioner’s combination of Kester and Honig does not disclose or suggest a mask that defines acceptable behavior for the object. (EX-1001 cls. 14, 28.)

150. With respect to the claimed “mask,” Petitioner asserts that Kester’s determination of expected network activity—i.e., network attributes for inclusion in the hash/policy table—discloses a mask. (Petition 57.) This assertion forms the foundation of its combination and its arguments for other limitations. For example, Petitioner relies on Kester’s use of the hash/policy table “to ensure that the applications continue to operate in the predetermined manner.” (Petition 61.) And Petitioner asserts that Kester teaches categorization of applications that are not operating in a predetermined manner—i.e., where the activity has not been previously reviewed as part of the hash/policy table. (Petition 61-63.) That is, Petitioner contends that the human reviewer’s determination of expected network activity for the hash/policy table allegedly constitutes the claimed mask.

151. However, Kester’s expected network activity—contained within its hash/policy table—is not a mask that “defines acceptable behavior” because the hash/policy table and the expected network activity identified within that table includes unacceptable behavior. (*See* EX-1004 [0070], [0185].)

152. For example, Kester teaches that when an application is not operating in a “predetermined manner,” it is placed in the uncategorized applications database for future review, such that those attributes might be added to the hash/policy table. (EX-1004 [0070].) That is, whether an application is operating in a “predetermined” or “expected” manner is dependent on whether that activity is contained within the

hash/policy table and not whether that activity is acceptable or unacceptable. Petitioner's discussion of the combination does not address this deficiency in any way, let alone provide any explanation for why a skilled artisan would have found this limitation obvious in light of both references. Thus, Petitioner has failed to show that the combination of Kester and Honig would have rendered the independent claims obvious. (EX-1001 cls. 14, 28.)

153. As the summaries above make plain, the invention claimed in the '389 patent is starkly different from Kester's approach. *See* Sections II(B), III(A), *supra*. The invention of the '389 patent generates a mask that defines only acceptable object behavior, thereby enabling the system to safely run and monitor an object based on whether it is operating within the boundaries of the mask. *See* Section II(B), *supra*. Kester does not teach or suggest this approach. Rather, Kester embodies the same problem that the '389 patent sought to overcome—*i.e.*, the need for a human administrator to review applications or network access requests to determine whether each request or application launch should be permitted or not. (EX-1004, Fig. 20 (“Human reviewer uses hints and web information to research network access by application. Using evidence from hints and other research, determine allowed behavior for each application.”), [0089]-[0090], [0187].)

154. Claims 14 and 28 rely on a mask that defines the boundaries of an object's behavior by defining acceptable behavior. (EX-1001 cl. 14 (“generating a

mask for said computer object that defines acceptable behaviour for the computer object, monitoring an operation of the computer object on at least one of the first and second remote computers and reclassifying the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.”) cl. 28 (“generate a mask for a computer object that is initially classed not as malware, said mask defining acceptable behaviour for the computer object”).) The invention limits the mask to acceptable behavior, such that future object behavior can be allowed or disallowed based on whether it falls within or outside of the mask. (Section II(B), *supra*.) The benefits of using a generated mask that defines acceptable behavior are that: (1) the invention avoids the delay associated with waiting for a human to review each object and designate whether specific behaviors should be permitted or not; and (2) when an object’s behavior extends beyond the mask, the invention knows that the object should be reclassified from “not malware” to “malware.”

155. Kester fails to disclose generating a mask and fails to disclose a mask that defines only acceptable behavior. As discussed above, the challenged claims recite generating a mask and using a mask that defines acceptable behavior to allow or disallow an object to run. In contrast, Kester relies on a human to pre-review each application and/or behavior and designate whether each application/behavior should be permitted or not before the hash/policy table may be used to monitor it. (*Id.*; EX-2021 at 8:3-20.) When Kester’s system encounters a new behavior that has not yet

been reviewed, that behavior is logged to a database to eventually be reviewed by a human reviewer. (EX-1004 [0059], [0144]-[0145].)

156. To argue that Kester discloses a mask that defines acceptable object behavior, Petitioner asserts that Kester discloses “expected or allowed network activity (also referred to as expected or predetermined behavior).” (Petition 57.) But Petitioner wrongly equates “expected” network activity with “allowed network activity.” Kester teaches that expected network activity may be allowed or disallowed based on the human reviewer’s determination. (E.g., EX-1004 [0185].) Specifically, Kester teaches that because “network attributes associated with the application may be different for each attempted access,” its system “may allow a first combination of one or more network attributes while disallowing a second combination of the one or more network attributes.” (EX-1004 [0185], [0083].) Kester describes network activity as “expected” or “predetermined” to indicate that a human reviewer has previously reviewed the specific combination of network attributes and that a predetermined association with a policy (to allow or not allow) has been stored in the hash/policy table. (EX-1004 [0057]-[0059] (“If the execution launch detection module...does not find the application hash in the hash/policy table [] (for example, the application is uncategorized or the application is behaving unexpectedly)...”).)

157. That is, an application operating in an expected or predetermined manner—*i.e.*, within the boundaries of Petitioner’s alleged “mask”—might still be

disallowed from continuing to run based on the associated policy. This result is intuitive based on Kester's goal—*i.e.*, “to manage the broader intersection of employees with their computing environments” on a workstation-by-workstation basis. (EX-1004 [0006], [0115]-[0116].) Within this context, Kester's system seeks to provide a different type of protection that is distinct from that of the '389 patent.

158. For example, an employer might wish to limit particular activity to management only. Following Kester's approach, a human administrator reviews the application and associated network access data. (*Id.*) The human administrator then determines that, where the network access attributes indicate that a manager's workstation has made the request, the request should be associated with a policy that allows the request. (*Id.*) But if a non-manager's workstation makes the request, the human administrator determines that such attributes should be associated with a policy that does not allow the request. (*Id.*) Kester teaches that a hash/policy table will be generated specific to the requesting workstation and download it to the workstation. (EX-1004 [0116].) Thus, while the particular activity, *e.g.*, the network access request, may be expected or predetermined such that it is included in the hash/policy table, it may be unacceptable and will not be allowed. (*Id.*)

159. Accordingly, Kester relies on a hash/policy table containing “expected” activity reviewed by a human administrator that may include acceptable activity but that also includes unacceptable activity. Under the district court's claim

construction—which Petitioner urged in the parallel litigation—the claimed mask must define only acceptable behavior, and acceptable behavior cannot be defined by the inverse of unacceptable behavior. *See* Section II(C), *supra*. Thus, Kester does not disclose a mask that defines acceptable behavior of the object.

160. As a result of this critical distinction, Kester’s “expected network activity” cannot provide the same benefits as the ’389 patent’s claimed mask. For example, the ’389 patent allows an object to run when its monitored behavior falls within the object’s mask:

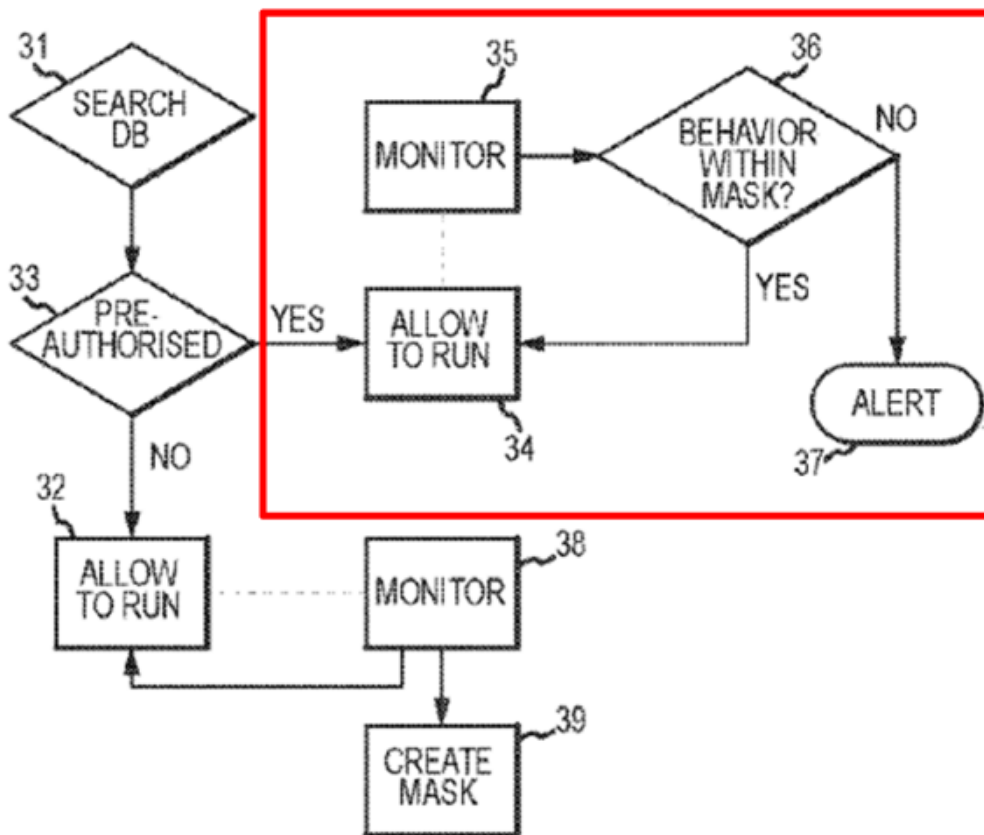


FIG.3

(EX-1001 Fig. 3 (annotated); *id.* cls. 14, 28.) And when that behavior extends beyond the mask, the object is not allowed to run and is reclassified as malware. (EX-1001, cls. 14, 28.) Petitioner expressly acknowledged this in its district court briefing. (EX-2022 10-11.) But because Kester’s expected network activity contains both acceptable and unacceptable activity, an application whose behavior is within this alleged mask cannot categorically be allowed to run. (EX-1004 [0042]-[0043], [0185]-[0186].) To do so would contradict Kester’s purpose, which is to apply predetermined policies on a workstation-by-workstation basis across all expected activity, thereby controlling employees’ interaction with the internet. (EX-1004 [0006], [0042]-[0043].)

161. As another example, because Kester’s expected network activity defines unacceptable behavior, it cannot be used as a tool for informing when an application must be reclassified as malware. Within the context of Kester’s disclosure, when an application is operating unexpectedly—*i.e.*, its hash does not match an existing hash in the hash/policy table—that merely means that the application or the application’s particular activity has not yet been reviewed, categorized, or associated with a policy by an administrator. (EX-1004 [0059].) Thus, the application—whether previously categorized or uncategorized—is added to the “uncategorized applications database” and subsequently reviewed by an administrator. (EX-1004 [0070]-[0071].) However, the fact that the activity-in-question was unexpected, *i.e.*,

previously unreviewed, does not mean that it is necessarily malicious or that the application should be categorized as malicious. Again, a human administrator will be required to review the application and the network attributes associated with its activity to determine whether such activity should be permitted or not, as well as whether the application should be categorized and/or recategorized. (EX-1004 [0070]-[0071].)

162. Because Kester's hash/policy table includes unacceptable expected network activity and because a human must determine expected network activity, Kester's teachings do not practice or suggest the challenged claim limitations, and a POSITA could not achieve the same benefits of the claimed invention. Accordingly, Petitioner has not shown (and cannot show) that Kester discloses or suggests generating a mask that defines acceptable behavior as recited in claims 14 and 28 claims of the '389 patent.

163. While Petitioner asserts that a POSITA would have been motivated to automate the human reviewer's determination of expected network activity in Kester's system based on Honig's teachings (Petition 58), this does not address the deficiency already present in Kester. Even if a POSITA were motivated to attempt to automate Kester's determination of expected network activity (which he would not be, as discussed further below), the discussion above shows that Kester's expected network activity is not a mask that defines acceptable behavior as claimed. *See*

Section III.A, *supra*. Thus, automatically generating Kester's expected network activity still would not practice the method or achieve the same benefits of the challenged '389 patent claims. Petitioner does not address this deficiency or argue that the prior art would have rendered such a mask obvious. Accordingly, Kester and Honig do not render any of the challenged claims obvious.

IX. CLAIMS AND/OR LIMITATIONS NOT ADDRESSED IN THIS DECLARATION

164. It is my understanding that this declaration is being provided in support of Patent Owner's Response. I have provided analysis of key limitations not included in the prior art. The fact that I have not analyzed particular limitations or claims in no way indicates or supports that these limitations or claims are disclosed or rendered obvious by the prior art, and I reserve the right to supplement my analysis if asked to do so.

X. SECONDARY OBJECTIVE INDICIA OF NONOBVIOUSNESS

165. I understand that in determining obviousness of the claimed invention, objective evidence of secondary considerations pertaining to nonobviousness must be considered, including long-felt but unsolved needs, failure of others, commercial success, etc. (*In re Huang*, 100 F.3d 135, 139 (Fed. Cir. 1996) (citing *Graham v. John Deere Co.*, 383 U.S. 1, 17-18 (1966).) I further understand that these indicia constitute independent evidence of nonobviousness that can be the most probative evidence in that regard "in avoiding the trap of hindsight when reviewing, what

otherwise seems like, a combination of known elements.” (*Leo Pharm. Prods., Ltd. v. Rea*, 726 F.3d 1346, 1358 (Fed. Cir. 2013).)

166. As demonstrated by my summary of the prior art, above, there was at least a long-felt and persistent need prior to the '389 patent to improve upon the time-consuming methods for determining whether computer objects are safe or malware. (*See* Section III, *supra*.) Prior attempts to develop such methods were limited in their ability to make rapid determinations and required extensive manual or semi-manual detailed analysis of objects and/or human administrator decisions. (*See* Section III, *supra*.) Where others failed, the '389 Patent succeeded. (*See* Section IV(A), *supra*.)

167. Thus, these secondary indicia only further support my opinion that the '389 Patent was non-obvious.

XI. CONCLUSION

168. 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 '389 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 29th day of March 2024 in Los Angeles.

Respectfully submitted,



Nenad Medvidovic, Ph.D.