

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.
Petitioner

v.

OPEN TEXT INC.
Patent Owner

Inter Partes Review No. 2023-01459

**PETITION FOR INTER PARTES REVIEW
OF U.S. PATENT NO. 8,726,389**

By: Robert M. Hansen
Matthew J. Anderson
Scott B. Amankwatia
The Marbury Law Group, PLLC
11800 Sunrise Valley Drive
15th Floor
Reston, VA 20191
Tel: (703) 391-2900
Fax: (703) 391-2901

TABLE OF CONTENTS

I.	Introduction.....	2
II.	Summary of the '389 Patent.....	2
A.	Description of the '389 Patent.....	2
B.	Summary of the Prosecution History	3
C.	The Advanced Bionics Framework Favors Institution	4
D.	Claim Construction Under 37 C.F.R. § 42.104(b)(3)	7
E.	Level of Ordinary Skill in the Art	8
III.	REQUIREMENTS UNDER 37 C.F.R. § 42.104	8
A.	Grounds for Standing (37 C.F.R. § 42.104(a))	8
B.	Identification of Challenge (37 C.F.R. § 42.104(b))	8
IV.	OVERVIEW OF THE PRIOR ART.....	9
A.	Kester	9
B.	Kennedy	11
C.	Honig	13
V.	THE CHALLENGED CLAIMS ARE UNPATENTABLE	14
A.	Ground 1: <i>Kester</i> in view of <i>Kennedy</i> Renders Claims 1-13, 15-27, 29-30 Obvious... 15	
1.	Claim 1	15
	1[P]. A method of classifying a computer object as malware, the method comprising:	
	 15	
	[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;	15
	[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;	21
[1.3] storing, at the base computer, said data received from the first and second remote computers;	22
[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;.....	23
[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.....	26
[1.6] classifying, by the base computer, the computer object as malware on the basis of said comparison.	34
2. Claim 2	35
2. A method according to claim 1, wherein the relationship is between objects related directly, or between objects related via other objects.	35
3. Claim 3	36
3. A method according to claim 1, wherein where an object and other objects are found to have a relationship, monitoring those related objects as a homogenous entity and identifying a common object of the homogenous entity as a malware object.	36
4. Claim 4	37
4. A method according to claim 1, comprising deducing an object to be malware based on the behaviour of a related object.	37
5. Claim 5	39
5. A method according to claim 1, wherein if at least one other object to which said computer object is related is classed as malware, then classifying said computer object as malware.....	39
6. Claim 6	39
6. A method according to claim 1, wherein said other objects include the object or similar objects stored on at least one of the first and second remote computers.....	39
7. Claim 7	41

7. A method according to claim 1, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said computer object.....	41
8. Claim 8	42
8. A method according to claim 1, wherein the data about the computer object that is sent from the respective remote computer to the base computer and that is used in the comparison includes one or more of: executable instructions contained within or constituted by the computer object; the size of the computer object; the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object;	42
the creation and modification dates of the computer object; vendor, product and version and any other information stored within the computer object; and the computer object header or header held by the respective remote computer.	42
9. Claim 9	43
9. A method according to claim 1, wherein the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the respective remote computer.	43
10. Claim 10.....	43
10. A method according to claim 9, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.	43
11. Claim 11.....	44
11. A method according to claim 9, wherein the key has at least one component that represents data about said computer object.....	44
12. Claim 12.....	45
12. A method according to claim 11, wherein said data about said computer object includes at least one of: the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object; the creation and modification dates of the computer object; vendor, product and version and any other information stored within the computer object; the computer object header or header held by the respective remote computer; and, events initiated by or involving the computer object when the computer object is created, configured or runs on the respective remote computers.	45
13. Claim 13.....	45

13. A method according to claim 9, wherein the key has at least one component that represents the physical size of the computer object.....	45
14. Claim 15.....	45
15[P]. An apparatus for classifying a computer object as malware, the apparatus comprising:	45
[15.1] a base computer constructed and arranged to receive 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;	46
[15.2] the base computer being constructed and arranged to receive data about the computer object from a second remote computer on which the computer object or similar computer objects re [sic] 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;	46
[15.3] the base computer being constructed and arranged to correlate 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;	46
[15.4] the base computer being constructed and arranged to compare the 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.....	47
[15.5] the base computer being constructed and arranged to classify the computer object as malware on the basis of said comparison.	47
15. Claim 16.....	47
16. Tha [sic] apparatus according to claim 15, wherein the base computer is constructed and arranged so as to identify the relationship between objects related directly, or between objects related via other objects.	47
16. Claim 17.....	47

17. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as, where an object and other objects are found to have a relationship, to monitor those related objects as a homogenous entity and identify a common object of the homogenous entity as a malware object.....	47
17. Claim 18.....	47
18. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to deduce an object to be malware based on the behaviour of a related object.	47
18. Claim 19.....	48
19. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.....	48
19. Claim 20.....	48
20. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least one of the first and second remote computers.	48
20. Claim 21.....	48
21. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.	48
21. Claim 22.....	49
22. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to compare the data where the data includes one or more of: executable instructions contained within or constituted by the computer object; the size of the computer object; the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object; the creation and modification dates of the computer object; vendor, product, and version and any other information stored within the computer object; and the computer object header or header held by the respective remote computer.	49
22. Claim 23.....	49
23. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on said respective remote computer.	49

23.	Claim 24.....	50
	The apparatus according to claim 23, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.....	50
24.	Claim 25.....	50
	25. The apparatus according to claim 23, wherein the key has at least one component that represents data about said computer object.....	50
25.	Claim 26.....	50
	26. Apparatus according to claim 25, wherein said data about said object includes at least one of the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the comptuer [sic] object; the creation and modification dates of the computer object; vendor, product, and version and any other information stored within the computer object; the computer object header or header held by the respective remote computer, and events initiated by or involving the computer object when the computer object is created, configured, or runs on the respective remote computers.	50
25.	Claim 27.....	51
	27. The apparatus according to claim 23, wherein the key has at least one component that represents the physical size of the computer object.....	51
26.	Claim 29.....	51
	29. A computer program recorded on non-transitory computer readable medium comprising program instructions for causing a computer to perform the method of claim 1.	51
27.	Claim 30.....	52
	30. A method according to claim 1, wherein the data received about the computer object from the first remote computer is different than the data received about the computer object from the second remote computer.....	52
B.	Ground 2: Kester in view of <i>Kennedy</i> and <i>Honig</i> Renders Claims 14 and 28 Obvious	53
1.	Claim 14	53
	14. A method according to claim 1, comprising initially classifying a computer object as not malware, 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.....	53
2. Claim 28	69
14. The apparatus according to claim 15, wherein the base computer is constructed and arranged to generate a mask for a computer object that is initially classed not as malware, said mask defining acceptable behaviour for the computer object, and the base computer is constructed and arranged to monitor an operation of the computer object on at least one of the first and second remote computers and to reclassify the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.....	69
VI. THE FINTIV AND GENERAL PLASTIC FACTORS FAVOR INSTITUTION	70
A. Stay	70
B. Proximity of the Court’s Trial Date	70
C. Investment in Parallel Proceeding.....	71
D. Overlap.....	71
E. Same Party.....	72
F. Other Circumstances	72
G. General Plastic Factors.....	73
VII. CONCLUSION	74
VIII. Mandatory Notices (37 C.F.R. § 42.8(a)(1)).....	75

TABLE OF AUTHORITIES

Cases

<i>Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH</i> , IPR2019-01469, Paper 6 (Feb. 13, 2020)	4, 6
<i>Apple Inc. v. Fintiv, Inc.</i> , IPR2020-00019, Paper 11 at 2–3, 6 (PTAB Mar. 20, 2020) (precedential)	71
<i>Apple Inc. v. Uniloc 2017 LLC</i> , IPR2018-00580, Paper 13 at 10-11 (Aug. 21, 2018)	74
<i>Becton, Dickinson & Co. v. B. Braun Melsungen AG</i> , IPR2017-01586, Paper 8, 17-18 (Dec. 15, 2017)	4
<i>Cisco Sys., Inc. v. Ramot at Tel Aviv Univ. Ltd.</i> , IPR202000122, Paper 15 at 10-11 (PTAB May 15, 2020)	73
<i>General Electric v. Vestas</i> , IPR2018-00928, Paper 9, *16-17 (PTAB Nov. 5, 2018)	7
<i>General Plastic Industrial Co., Ltd. v. Canon Kabushiki Kaisha</i> , IPR2016-01357, Paper 19 at 16 (PTAB Sept. 6, 2017)	74
<i>Hewlett Packard Enterprise Co., et al. v. Billjco LLC</i> , IPR2022-00420, Paper 17, at 6 (PTAB July 12, 2022)	73
<i>NHK Spring Co., Ltd. v. Intri-Plex Technologies, Inc.</i> , IPR2018-00752, Paper 8 at 19–20 (PTAB Sept. 12, 2018) (precedential)	71
<i>Phillips v. AWH Corp.</i> , 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc)	7
<i>Sand Revolution II, LLC v. Continental Intermodal Group-Trucking LLC</i> , IPR2019-01393, Paper 24 at 7 (PTAB Jun. 16, 2020)	71, 72
<i>Shenzhen Carku Tech. Co., Ltd. v. The Noco Co.</i> , IPR2020-00944, Paper 20 at 61 (PTAB Nov. 12, 2020)	71
<i>Thorne Research, Inc. v. Trustees of Dartmouth College</i> , IPR2021-00491, Paper 18, 7-9 (PTAB Aug. 12, 2021)	6
<i>Western Digital Corporation v. Spex Technologies, Inc.</i> , IPR2018-00084, Paper 14, *11 (PTAB Apr. 25, 2018)	7

Statutes

35 U.S.C. § 314(a)	71
--------------------------	----

Rules

37 C.F.R. § 42.10(a)..... 77

37 C.F.R. § 42.104(a)..... 8

37 C.F.R. § 42.104(b) 8

37 C.F.R. § 42.8(a)(1) 76

37 C.F.R. § 42.8(b)(2)..... 76

37 C.F.R. § 42.8(b)(3)..... 77

I. Introduction

Trend Micro, Inc. (“Petitioner”) petitions for *inter partes* review of claims 1-30 of U.S. Patent No. 8,726,389 to Morris *et al.*, entitled “METHODS AND APPARATUS FOR DEALING WITH MALWARE” (“the ’389 patent,” EX1001), which according to PTO records is assigned to Webroot, Inc.

II. Summary of the ’389 Patent

A. Description of the ’389 Patent

The ’389 Patent generally relates to methods and systems for classifying a computer object as malware. ’389 Patent (EX1001), Abstract. The ’389 Patent describes a community-based approach that compares objects from multiple remote computers:

[In a base computer,] receiving data about a computer object from each of plural remote computers on which the object or similar objects are stored; such data may be stored in [the base computer to] provide[] an extensive corollary of an object’s creation, configuration, execution, behavior, identities, and relationships to other objects that either act upon it or are acted upon by it. The stored data may then be compared to identify relationships between the obje[c]t and other objects or entities. Based on this comparison, the computer object may be classified as malware.

’389 *File History* (Ex. 1002), 141-142; *see also* ’389 *Patent*, 1:9-47, 3:7-24, Fig. 1 (depicting community-based architecture).

In particular, the alleged invention “utilises [sic] an object’s relationship to

other objects that act upon it or upon which it acts. For example, analysis can be made of which object created this object, which objects this object created...” ’389 *Patent*, 11:5-11. This “relationship chain provides a complex series of correlation points, allowing ancestral relationships to be considered for any object and its predecessors by event or its issue[.]” *Id.*, 11:13-17. These relationships “can be used immediately and automatically to mark a child object as bad (or good) if the or a parent or other related object is bad (or good).” *Id.*, 3:38-41.

B. Summary of the Prosecution History

The ’389 Patent was filed on July 8, 2012 as a continuation of U.S. Patent No. 8,418,250 filed June 30, 2006 and claims priority to a foreign application filed June 30, 2005. For this proceeding, Petitioner applies June 30, 2005, as the priority date for the Challenged Claims.

Then-pending independent claims 1 and 15 of the ’389 Patent’s application were ultimately rejected during prosecution over U.S. Pub. No. 2006/0085857 to Omote et al. (“*Omote*”) in view of U.S. Pub. No. 2003/0177394 to Dozortsev et al. (“*Dozortsev*”). ’389 *File History* (EX1002), 155-165. The Examiner found this combination taught the claimed receipt of event data from multiple remote computers; comparison of this data with other objects to identify a relationship; and object classification based on said comparison. *Id.*

Applicant amended its claims to require, in relevant part, “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” and “comparing by the base computer the correlated data...” *Id.*, 181.¹ Applicant argued the cited art failed to teach this correlating step along with the comparison of this correlated data to other objects or entities. *Id.*, 189, 191. The Examiner agreed and issued a Notice of Allowance citing the same as the basis for allowance. *Id.*, 210.

As detailed below, *Kester* in view of *Kennedy* and *Honig* discloses the claimed community-based anti-malware methods and systems including the recited “correlating” and “comparison” steps/functions.

C. The Advanced Bionics Framework Favors Institution

Under the two-part Advanced Bionics framework applying the Becton, Dickinson factors, part one considers whether the same or substantially the same prior art or arguments were previously presented to the office—Becton factors (a), (b), and (d). *Advanced Bionics, LLC v. MED-EL Elektromedizinische Geräte GmbH*, IPR2019-01469, Paper 6 (Feb. 13, 2020); *Becton, Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8, 17-18 (Dec. 15, 2017):

¹ All emphasis added by Petitioner unless noted otherwise.

(a) the similarities and material differences between the asserted art and the prior art involved during examination;

(b) the cumulative nature of the asserted art and the prior art evaluated during examination;

(d) the extent of the overlap between the arguments made during examination and the manner in which Petitioner relies on the prior art or Patent Owner distinguishes the prior art[.]

The first part of this framework is not satisfied and discretionary denial under § 325(d) is not appropriate.

The Office never previously considered the instant Grounds, which rely on U.S. Patent Pub. No. 2005/0210035 to Kester et al. (“*Kester*”) in combination with U.S. Patent Nos. 7,594,272 to Kennedy et al. (“*Kennedy*”) and 7,225,343 to Honig et al. (“*Honig*”). During prosecution, *Kester* was advanced in an office action as a tertiary reference in combination with *Omote* and *Dozortsev* to teach “generating a mask for said object that defines acceptable behaviour for the object, and monitoring operation of the object on at least one of the remote computers and reclassifying the object as malware if the actual monitored behaviour extends beyond that permitted by the mask[,]” found in dependent claims 14 and 28. ’389 *File History*, 163-164. *Kennedy* and *Honig* were neither cited nor considered during prosecution.

The Proposed Grounds herein rely on *Kester* as a primary reference in combination with the newly proposed *Kennedy* and *Honig* references. Because the Proposed Grounds rely on *Kester* in a fundamentally different way than it was analyzed during prosecution and because they include teachings from *Kennedy* and *Honig* that were not considered during prosecution, the instant grounds are materially different and non-cumulative to the rejections evaluated during examination.

The instant grounds are significantly different from the *Omote/Dozortsev* combination used in prosecution which failed to address the amended “correlation” step and comparison of correlated data. For at least these reasons, the Office cannot be deemed to have considered the “same or substantially the same” art or arguments within the meaning of Section 325(d) and the first prong of the *Advanced Bionics* framework should not be used to deny institution, ending the inquiry. *Thorne Research, Inc., v. Trustees of Dartmouth College*, IPR2021-00491, Paper 18, 7-9 (PTAB Aug. 12, 2021) (“[t]he fact that one piece of art from the combination was previously presented and/or argued to the Office alone is insufficient to satisfy the first prong of *Advanced Bionics*[,]” granting institution where non-cited/considered references were presented in combination with a considered reference and refusing to consider part two of *Advanced Bionics* on this basis).

D. Claim Construction Under 37 C.F.R. § 42.104(b)(3)

In this proceeding, claims are interpreted under the same standard applied by Article III courts (i.e., the *Phillips* standard). 37 C.F.R. § 42.100(b); *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312 (Fed. Cir. 2005) (en banc).

In parallel litigation, PO argues all claim terms should be given their plain and ordinary meanings. EX1007 (Dkt. 98, *PO Opposition Brief*), 4-22. For the purposes of this proceeding only, Petitioner proposes the Board adopt PO's proposal. *See Western Digital Corporation v. Spex Technologies, Inc.*, IPR2018-00084, Paper 14, *11 (PTAB Apr. 25, 2018) (finding Petition based on claim constructions urged by Patent Owner satisfies the claim construction requirements and Petitioner is not required to express its subjective agreement regarding correctness of the proffered claim construction or take ownership of the construction)); *see also General Electric v. Vestas*, IPR2018-00928, Paper 9, *16-17 (PTAB Nov. 5, 2018).²

² Petitioner reserves the right in other proceedings to construe the Challenged Claims and to argue they are indefinite.

E. Level of Ordinary Skill in the Art

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. Declaration of Dr. Lee ("Decl.") (EX1003), ¶¶34-37.

III. REQUIREMENTS UNDER 37 C.F.R. § 42.104

A. Grounds for Standing (37 C.F.R. § 42.104(a))

Petitioner certifies that the '389 patent is available for IPR.

B. Identification of Challenge (37 C.F.R. § 42.104(b))

Petitioner requests the Challenged Claims be found unpatentable on the following grounds:

Proposed Grounds of Unpatentability	Exhibits
<u>Ground 1:</u> Claims 1-13, 15-27, 29-30 are obvious under §103(a) over <i>Kester</i> in view of <i>Kennedy</i>	Ex. 1004 Ex. 1005
<u>Ground 2:</u> Claims 14 and 28 are obvious under §103(a) over <i>Kester</i> in view of <i>Kennedy</i> in further view of <i>Honig</i>	Ex. 1004 Ex. 1005 Ex. 1006

IV. OVERVIEW OF THE PRIOR ART

A. Kester

Kester was filed on May 19, 2005, published September 22, 2005, and qualifies as prior art under 35 U.S.C. §102(e) (pre-AIA). *Kester*, (EX1004).

Kester describes methods for categorizing an application running on a remote workstation (e.g., as malware) based on the application's behavior. *Kester*, [0069][0071], [0082]. Specifically, *Kester* teaches collecting behavior data, i.e., “collection data,” from multiple remote workstations, and using that data to categorize the application. *Kester*, [0073]-[0076]. *Kester*'s methods are implemented over a community-based architecture:

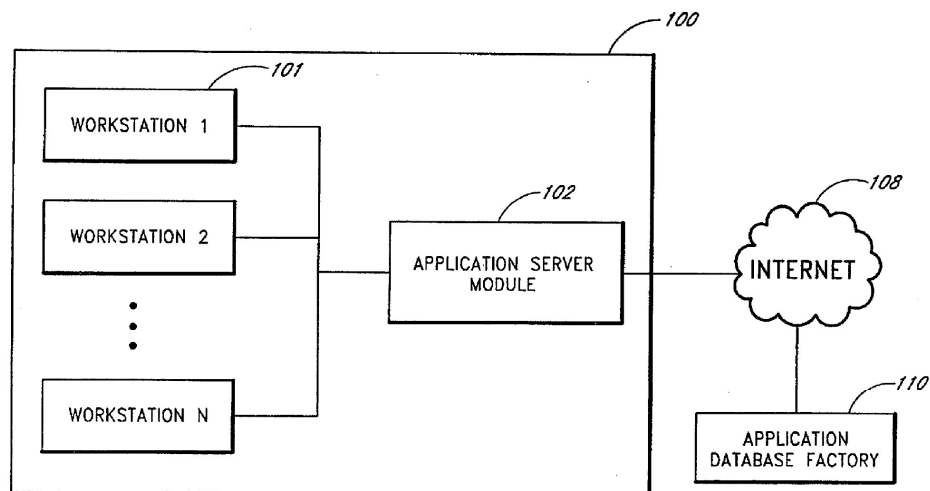


FIG. 1

Id., Fig. 1.

Workstations 101 may be “personal computer[s]” (*id.*, [0037]) and include a workstation management module 200 which “can detect the launch of an application on the workstation 101” and/or “monitor the ongoing network activity or behavior of the application.” *Id.*, [0040]-[0041]. Collection data is collected during these processes and sent to an application server module 102 for processing. *Id.*, [0051], [0064], [0069], [0074]. Processing includes “merging and sorting the collection data from multiple workstation management modules.” *Id.*; *see also id.*, [0112]. Application server module 102 can then compare this collection data to “expected network access data for the application.” *Id.*, [0069]. This module “monitor[s] the behavior of categorized and uncategorized applications[,]” and when a categorized application “does not operate in a predetermined manner [it] is placed in the uncategorized applications database 108.” *Id.*, [0069]-[0070].

Applications and associated data placed in the uncategorized applications database are accessed by “[t]he network administrator, via the classification user interface 106, [which] is then able to categorize the uncategorized application[.]” *Id.*, [0076]; *see also id.*, [0082] (listing categories and parent groups, including “malicious applets and scripts” and “malware”), Fig. 4A (depicting same). “[I]f the network administrator does not classify the application, the application database factory 110 can classify the collection data.” *Id.*, [0076]. A similar process occurs at the application database factory—a reviewer can use the analyst classification

module 302 preprogrammed with “algorithmically derived hints” to determine the appropriate categories for the application. *Id.*, [0087-0090], [0092].

According to the ’389 Patent, “the present invention relates to a method and apparatus for classifying a computer object as malware.” ’389 Patent, 1:10-12; *see also id.*, Claims 1, 15. *Kester* also provides a method and apparatus for classifying a computer object as malware. *Kester*, [0098]; *see also id.*, [0079], [0082], [0134], Fig. 4A (“Malware” parent group and “Malicious Applets and Scripts” category); *see also* Ground 1, Element 1.6. Accordingly, *Kester* is in the same field of endeavor as the ’389 Patent. *Decl.*, ¶¶56-64. Therefore, *Kester* is analogous art to the ’389 Patent.

Kester is also reasonably pertinent to certain problems addressed by the ’389 Patent such as the need to initially classify an object as not malware, generate a mask for the object that defines acceptable behavior for the object, and reclassify the object if the actual monitored behavior extends beyond the mask. *Compare* ’389 Patent, 4:25-36, 14:61-15:10, Claims 14 and 28, with *Kester*, [0076], [0179]-[0181], Claims 1-2; *Decl.*, ¶¶56-64. Therefore, *Kester* is analogous art to the ’389 Patent.

B. Kennedy

Kennedy was filed on October 5, 2004, issued on September 22, 2009, and qualifies as prior art to the '389 Patent under 35 U.S.C. § 102(e) (pre-AIA).

Kennedy, (EX1005).

Kennedy is directed at detecting malicious software through file group behavior. *Kennedy*, Title, Abstract. *Kennedy*'s system uses a "malicious software detection module (MSDM)" executing on a computer system connected to a network to detect suspicious software and "track[] the set of files that are associated with the suspicious software." *Id.*, 1:42-49. The MSDM comprises a "file set tracking module 312" which "maintains a set of files for each suspicious software." *Id.*, 4:19-20; Fig. 3 (depicting components of MSDM). The "tracking module 312 tracks the behaviors of the files in the set and records each time a file in the set 'touches' another file[.]" *Id.*, 4:23-28. "The file set tracking module 312 [then] adds the touched file to the set of the file that touched it." *Id.*, 4:33-34.

The MSDM further includes a "file set behavior monitoring module 316" which "employs one or more heuristics to determine whether the suspicious software represented by each set is malicious." *Id.*, 5:3-11. "Each heuristic describes one or more conditions that, if satisfied, indicate that the software is malicious. A heuristic is satisfied if any file in the software's set fulfills the conditions of the heuristic, and/or the collective actions of the files in the set fulfill the conditions of the heuristic." *Id.*

Like the '389 Patent, *Kennedy* teaches means of detecting malicious software by tracking relationships of files. *Compare Kennedy*, 1:6-9, 4:19-28; *see also id.*, Abstract, Fig. 4 (step 418), with '389 Patent, 1:9-12, 3:37-47, 11:5-17, Claim 1-3, 15-17. Accordingly, *Kennedy* is in the same field of endeavor as the '389 Patent. *Decl.*, ¶¶65-68. Therefore, *Kennedy* is analogous art to the '389 Patent.

Kennedy is also reasonably pertinent to certain problems addressed by the '389 Patent such as the need for a rapid determination of the nature of an object, without requiring detailed analysis of the object itself or relying on signature matching. *Compare '389 Patent*, 1:23-34, 3:14-24, Claim 1, with *Kennedy*, 1:11-52; *Decl.*, ¶¶65-68. Therefore, *Kennedy* is analogous art to the '389 Patent.

C. Honig

Honig was filed on January 27, 2003, issued May 29, 2007, and qualifies as prior art under 35 U.S.C. §102(e) (pre-AIA). *Honig*, (EX1006).

Honig describes “an architecture to automate the processes of data collection, model generation and data analysis” to improve Intrusion Detection Systems (IDS). *Honig*, 4:53-56. *Honig* system comprises sensors 12 which “gather information from an environment and send that data to [a] data warehouse 14.” *Id.*, 8:14-24. “The data in the data warehouse 14 is accessed by [] detection model generators 16 which generate models that classify activity as either malicious or

normal.” *Id.* (describing using “audit data” as exemplary input to the models).

Honig’s “system 10 is designed to work with any model generation algorithm.” *Id.*

13:61-14:4. This includes, for example, “misuse detection,” “supervised (traditional) anomaly detection[,]” and “unsupervised anomaly detection[,]” *Id.*, 20:33-38.

Like the ’389 Patent, *Honig* teaches a method and system for classifying applications as normal or anomalous. *Compare Honig*, 15:43-16:24, with ’389 Patent, 1:9-12. Accordingly, *Honig* is in the same field of endeavor as the ’389 Patent. *Decl.*, ¶¶69-75.

Honig is also reasonably pertinent to certain problems addressed by the ’389 Patent such as the need to automatically generate (e.g., without human decisions) a mask for an object that defines normal/acceptable behavior for the object, and to classify the object as anomalous if the actual monitored behavior extends beyond the mask. *Compare ’389 Patent*, 4:27-36, 15:11-31, Claims 14, 28 with *Honig*, 8:15-18, 24:43-45; *see also id.*, 4:54-57; *Decl.*, ¶¶69-75. Therefore, *Honig* is analogous art to the ’389 Patent.

V. THE CHALLENGED CLAIMS ARE UNPATENTABLE

**A. Ground 1: Kester in view of Kennedy Renders Claims 1-13, 15-27,
29-30 Obvious**

1. Claim 1

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

To the extent the preamble is limiting, *Kester* in view of *Kennedy* renders obvious a method of classifying a computer object as malware. See Element 1.6 below.

[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 objects³ 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;

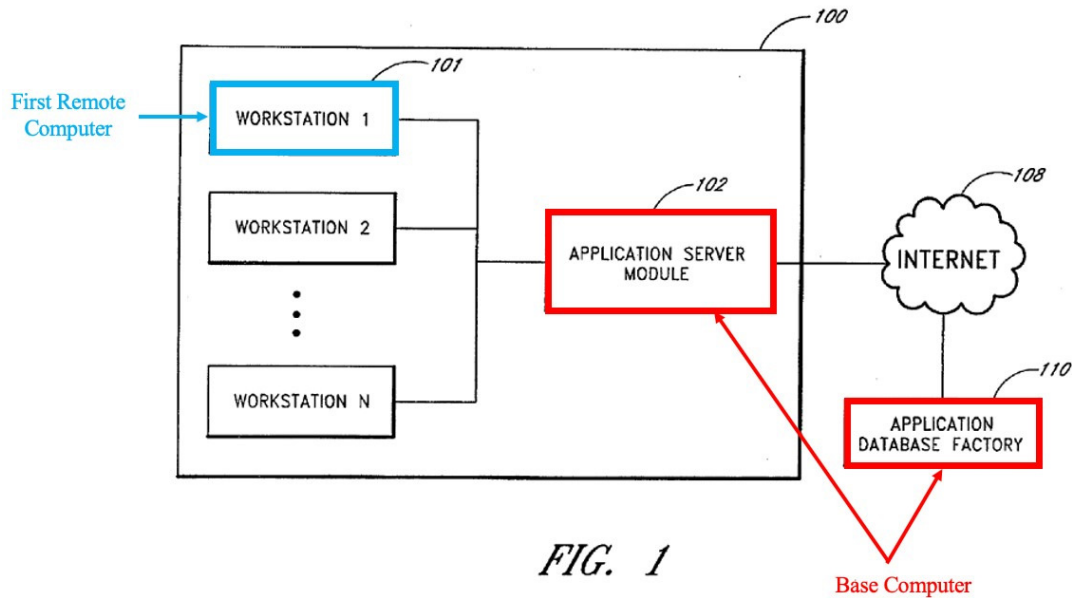
Kester teaches a base computer (application server module 102 and/or application database factory 110) that receives data about a computer object

³ Petitioner proposes the term “similar objects” is indefinite in litigation as the bounds of the term are unclear. EX1008 (*Defendants’ Opening Markman Brief*), 2-4. Regardless, the claims are still rendered obvious by art cited herein. Because the claim is written in the alternative, any uncertainty as to the boundaries of “similar objects” does not preclude

(program/application) from a first remote computer ([workstation 1](#)) on which the object is stored. *Kester*, [0008], [0036]-[0037], Claim 1 (“The system comprises a workstation configured such that a program resident thereon can access a network,...send program data associated with the program to an application server module[,]...and an application server module coupled to the workstation and configured to receive the program data from the workstation management module...if the program is not operating in a predetermined manner, then send the program data to an application database factory[.]”). *Kester’s* program/application is an “object” as described by the ’389 Patent. ’389 *Patent*, 1:19-22 (object includes “any executable computer file”), 8:1-4 (defining “object” broadly as “a computer file, part of a file or a sub-program, macro, web page or any other piece of code to be operated by or on the computer”).

Kester’s architecture is shown below:

application of prior art that discloses “receiving data about a computer object from a first remote computer on which **the computer object** [is] stored[.]”



Kester, Fig. 1 (annotated). The application server module and/or application database factory⁴ (*base computer*) receives data from each workstation. Namely, each workstation uploads application related data from its logging database 206 to the application server module 102 “immediate[ly] or periodic[ally].” *Kester*, [0064][0065], Fig. 2; *see generally id.*, [0073], [0020]-[0021], [0027]-[0028], Figs. 8, 9, 15, 16. The application server module 102 may then transmit this data to the application database factory 110 “immediate[ly] or periodic[ally].” *Kester*, [0078], Fig. 3; *see generally id.*, [0023-0024], [0030-0031], [0086]-[0087], [0132], [0176], Figs. 11, 12, 18, 19; Fig. 6 (depicting exemplary application related data received by and displayed at the application database factory).

⁴ Although the application database factory can comprise multiple computers, it can also comprise only “one” computer, i.e., the *base computer*. *Kester*, [0084].

Kester discloses the received data (*Kester*'s "application related data") includes information about events involving the application when the application runs on workstation 1. For example, when an application on a workstation is launched and/or requests access to the network (both of which qualify as "events"), the workstation logs application related data (e.g., properties/attributes associated with the requested application) in logging database 206. *Kester*, Fig. 14 (steps 1402, 1412, 1418); *see generally id.*, [0139]-[0146], Figs. 2, 14. *Kester* refers to application related data as "collection data." *Kester*, [0061]-[0062], [0073], [0051], [0099], [0141], Fig. 4B.

This "collection data" associated with *Kester*'s application launching or requesting access to the network includes "event" information as described by the '389 Patent. The '389 Patent defines events as "the actions or behaviours of an object acting upon another object or some other entity." '389 Patent, 13:3-4; *see also EX1007* (Dkt. 98, *PO Opposition Brief*), 15 ("the [] '389 specification[] [is] clear that 'events' include both 'actions' and 'behaviors'"). Like *Kester*, the '389 Patent provides an exemplary event involving a program connecting to the network. *Id.*, 12:24-28 ("program performing an event to connect with an Internet Chat Relay site"). Thus, when *Kester*'s application launches or requests network access, these are "events...involving the computer object when the computer

object...runs on the first remote computer” as described by the ’389 Patent. *Decl.*, ¶¶77-82.

Kester further discloses this “collection data” associated with the above events includes an identity of an object initiating the event and an identity of an object or other entity on which the event is being performed. Namely, *Kester*’s “collection data” comprises information about the application such as “application name” (i.e., identity of an object initiating the event), “source IP address” (i.e., when the application is launched, the source IP address identifies an entity on which the launch event is being performed) and “destination IP address” (i.e., when requesting access to the network, the destination IP address identifies an entity on which the access request event is being performed). *Kester*, [0061]-[0062], [0073]; *see also id.*, [0051], [0099], [0141], Fig. 4B; *Decl.*, ¶83.

Kester also collects and sends “additional information or properties [] associated with a launched or network accessing application[,]” which a PHOSITA would have understood to include an “event type[.]” *Kester*, [0051]; *Decl.*, ¶¶84-86. Indeed, *Kester*’s system must ascertain whether the application is launching or requesting access to the network (two “types” of events). *Kester*, [0137]-[0138] (Figs. 7-13 relate to launch events and Figs. 14-20 relate to network access events); *Decl.*, ¶¶84-86. These “event types” are consistent with the ’389 Patent’s disclosures. *Id.*; ’389 Patent, 12:21-28 (noting the database can store an unlimited

number of event types, such as “an event to connect with an Internet Chat Relay site” over a network).

Alternatively, it would have been obvious to a PHOSITA to additionally collect and send an “event type” along with the above information to further inform subsequent categorization of the application—this categorization process is described in detail in Element 1.g below. *Decl.*, ¶¶87-92. *Kester* discloses several sets of information collected relating to an application launch/network access request, including the “name, publisher, suite, hash, file size, version, protocol, I.P. address, port, and additional information or properties which are associated with a launched or network accessing application.” *Kester*, [0051], [0061] (further collecting “request frequency” and “time of execution”), [0073] (collecting “trace ID”). And *Kester* generally contemplates collecting “other network access data” (see *id.*) such that “the collection data can include any additional data associated with the applications[.]” *Id.*, [0108]. A PHOSITA, therefore, would have been motivated to modify *Kester* to collect additional data including an “event type” as *Kester* expressly contemplates. *Decl.*, ¶¶87-92.

More particularly, in *Kester*, collection data is gathered to “aid in categorizing the application.” *Kester*, [0097]-[0098]. Collecting an event type—a well-known parameter in computer security systems—would enhance *Kester*’s categorization process such that when the event type signifies the application is

launching or requesting network access, the application is behaving expectedly.

Decl., ¶¶89-90 (describing known process in *Farley* for collecting an “event type parameter” from an event). However, if an event type signifies an application is performing other actions, e.g., downloading files, modifying files resident on the workstation, etc., this would indicate unexpected behavior such that the application could be categorized or recategorized accordingly. *Decl.*, ¶¶87-92. Because *Kester* already contemplates collecting additional data, and collecting event type data was well-known in the field, there would have been a reasonable expectation of success making this simple modification to *Kester*’s data collection process. *Id.*, ¶91

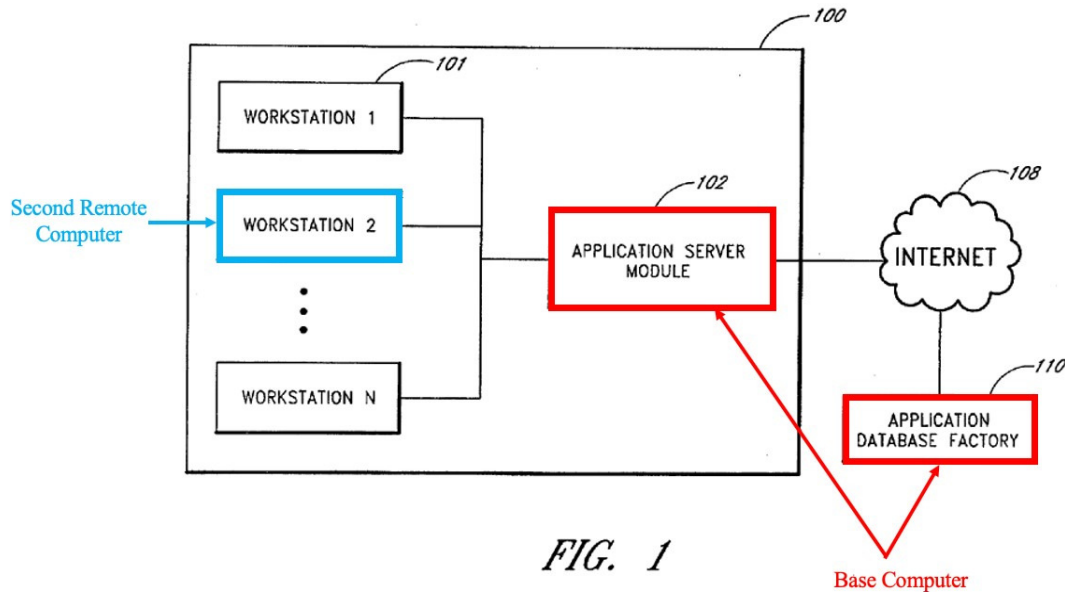
Thus, *Kester* discloses Element 1.1. *Decl.*, ¶¶87-92.

[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;

As explained in Element 1.1 above, *Kester*’s **application server module and/or application database factory** (*base computer*) receives “collection data” from each **workstation**. See Element 1.b. *Kester* provides that the same application can be stored on the plurality of workstations 101. *Kester*, [0157] (“different

workstations 101 can have...the same application running thereon”); *see also id.*, [0041]. Indeed, collection data can be uploaded from multiple (potentially thousands) of workstations. *Kester*, [0077], [0079], [0095], [0119], [0162].

Kester’s second workstation is depicted below:



Id., Fig. 1 (annotated). For the reasons above, a PHOSITA would have been motivated to additionally collect an “event type” along with the other collection data to inform *Kester*’s categorization process. *See* Element 1.1.

Thus, *Kester* discloses Element 1.2. *Decl.*, ¶¶93-95.

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

Kester discloses its application server module and/or application database factory (*base computer*) stores the received collection data from each workstation. *See* Elements 1.1 and 1.2. For example, *Kester* teaches that its application server

module 102 includes a “workstation upload/download module 104” (*see Kester*, Fig. 3) which “receives the collection data from the upload/download module 203” in the workstation. *Kester*, [0074]; *see also id.*, [0075]-[0077] (noting the collection data can further be stored in module 102’s application inventory database 103 in the uncategorized applications database 108 and/or network access database 107), [0112], Fig. 8 (step 830), Fig. 16 (step 1616).

Similarly, *Kester*’s application database factory stores this collection data. *See, e.g., Kester*, [0078] (“the factory upload/download module 105 is configured to transmit data from the application inventory database 103 to the application database factory 110”). Like module 102, factory 110 includes an “upload / download module 301” used to “receive collection data (for example, applications and any additional data associated with the application) from the application server module 102.” *Id.*, [0086]-[0087], [0132], Fig. 12 (step 1216 “receive collection data”), Fig. 18 (step 1818).

Thus, *Kester* discloses Element 1.3. *Decl.*, ¶¶96-98.

[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;

Kester discloses that its application server module and/or application database factory (*base computer*) correlate at least a portion of the collection data

received from the remote workstations. Namely, *Kester* discloses the “workstation upload/download module 104” in the application server module “receives the collection data from the upload/download module 203” in the workstation and “processes the collection data.” *Kester*, [0074]. This processing includes “merging and sorting the collection data from multiple workstation management modules.” *Id.*; see also *id.*, [0112], [0119], [0154], [0162]. The application server module’s merging step is depicted below:

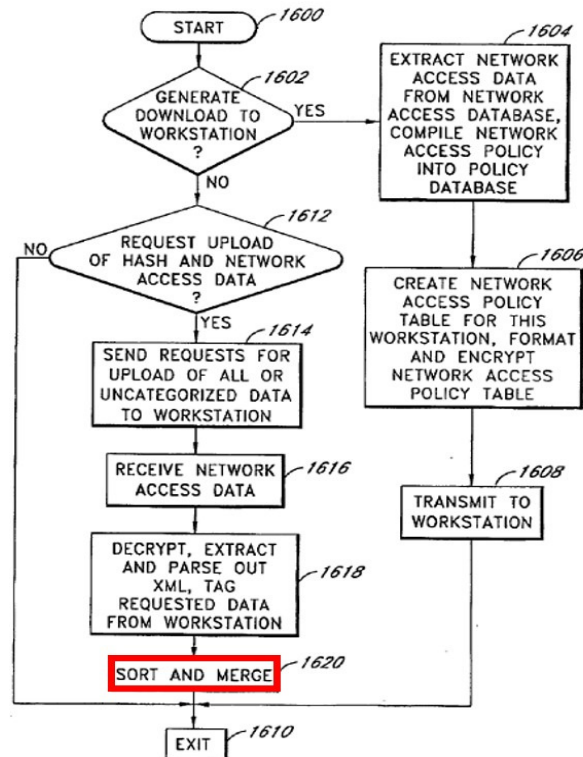


FIG. 16

Id., Fig. 16 (annotated).

Likewise, *Kester*’s application database factory comprises an “upload / download module 301” which “receives the collection data from the factory

upload/download module 105” in the application server module and “processes the collection data.” *Id.*, [0087]. This processing also includes “merging, sorting” the collection data. *Id.*; *see also id.*, [0133], [0177]. The application database factory’s merging step is depicted below:

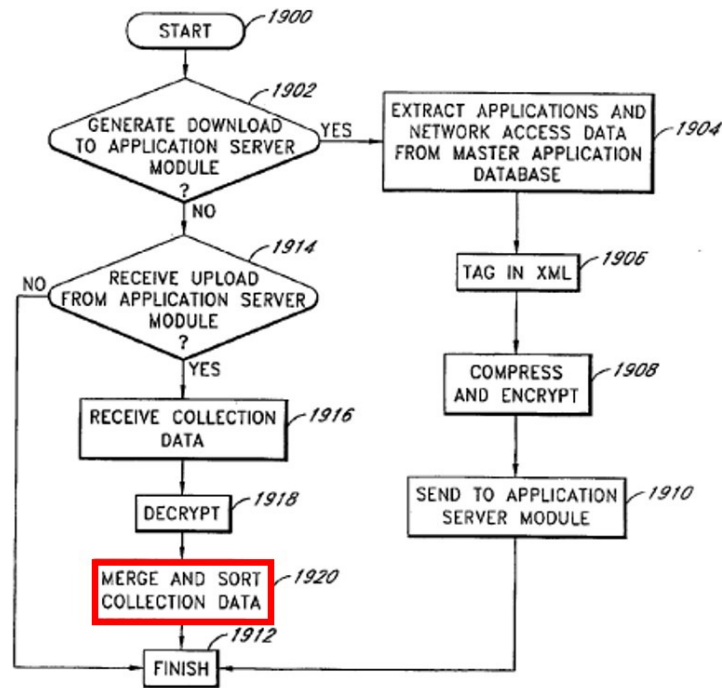


FIG. 19

Id., Fig. 19 (annotated); *see also id.*, Claims 21-22.

A PHOSITA would have understood *Kester*’s process of merging and sorting collection data from each workstation “correlates” that information as that term is used by the ’389 Patent. *Decl.*, ¶¶99-103. The specification broadly explains that data received from multiple remote computers may be stored in a centralized database which “provides an extensive corollary of an object’s creation,

configuration, execution, behaviour, identities and relationships to other objects...”

’389 *Patent*, 10:37-40, 11:55-59 (“The centrali[z]ation of the community data in the community database 7 provides the ability to rapidly assimilate object behaviours...”). Analogously, *Kester*’s process merges and sorts collection data into centralized databases from the multiplicity of workstations, as explained above, and therefore “correlates” this data in the same way as the ’389 *Patent*. *Decl.*, ¶¶99-103.

Thus, *Kester* discloses Element 1.4. *Decl.*, ¶¶99-104.

[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

In *Kester*, the application server module 102 and/or application database factory 110 receives collection data from each workstation, merges that data, and uses that merged collection data (i.e., *correlated data about the computer object received from the first and second remote computers*) to determine a group/category for the application. More particularly, *Kester* discloses each application file can be associated with one or more parent groups and/or categories. An exemplary parent group is “malware.” *Kester*, [0015], [0081]-[0082], [0098]. An exemplary category is “malicious applets and scripts”—i.e., malware. *Id.*, [0082]; *see also* ’389 *Patent*, 1:19-22 (defining “malware”); *Decl.*, ¶¶105-106. An

exemplary “malware” parent group and a “malicious applets and scripts” category is shown below in Figure 4A:

FIG. 4A

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

Kester, Fig. 4A (annotated).

To determine whether the merged collection data indicates the application should be categorized as malware, *Kester* derives expected network access data from applications behaving in a predetermined manner; the application server module 102 monitors “the behavior of categorized and uncategorized applications[;]” and a network administrator may then interface with the application server module 102 to “classify uncategorized applications and/or recategorize previously categorized applications,” (*see Kester*, [0070]-[0071]) by comparing “each application and any additional data that is associated with the application” to this expected activity “to determine its appropriate category or categories.” *Kester*, [0134], *see also id.*, [0119]-[0120] (describing merging collection data from multiple workstations, including the request frequency, and classifying the application based on this merged data); *see generally id.*, [0071], [0078], [0084], [0122], Figs. 3, 10; *see also id.*, Fig. 20 (describing similar categorization process at application database factory 110).

The “expected network activity” used to inform this categorization “can be determined from network activity by a different but related application.” *Kester*, [0044], [0182]. A PHOSITA would have understood making such a determination requires comparing the merged collection data to “other objects or entities” in the base computer. *Decl.*, ¶¶107-108.

Kester does not detail that identifying an application's relationship to another application is part of the process of determining whether received collection data indicates that an application is malware. However, *Kennedy* teaches this concept and, as explained below, a PHOSITA would have been motivated to implement *Kester* pursuant to *Kennedy*'s teachings such that when a familial relationship to known malware is discovered by comparing correlated collection data to data from other objects, the modified system automatically classifies the application as malware.

Kennedy teaches a malicious software detection module (MSDM) which "tracks the set of files that are associated with the suspicious software." *Kennedy*, Abstract. The MSDM includes a file set tracking module 312 that "maintains a set of files for each suspicious software" and "tracks the behaviors of the files in the set and records each time a file in the set 'touches' [(e.g., creates, modifies, and/or reads)] another file on the storage device 108 or elsewhere in the computer system 100." *Id.*, 4:19-33. Each touched file is added to the set of the file that touched it. *Id.*, 4:33-34. File sets and monitored behavior may be stored in a database. *Id.*, 4:4857. Alternatively, "a file's attributes can indicate that it is a member of a particular file set and/or associated with particular suspicious software" and that the tracking module 312 can update the attribute data "to reflect any activity by the suspicious software being monitored." *Id.*, 4:48-63.

Kennedy's behavior monitoring module 316 can then “monitor[] the behaviors of the files within the set associated with each suspicious software[,]” and “employ[] one or more heuristics to determine whether the suspicious software represented by each set is malicious.” *Kennedy*, 4:64-5:6; *see also id.*, 4:48-63 (noting a file’s attributes “can indicate that it is a member of a particular file set and/or associated with particular suspicious software”). As commonly known to a PHOSITA at the time, *Kennedy* specifies that “[e]ach heuristic describes one or more conditions that, if satisfied, indicate that the software is malicious.” *Id.*, 5:6-8; *Decl.*, ¶¶109-111. When tracking related files, *Kennedy* discloses: “[a] heuristic is satisfied if any file in the software’s set fulfills the conditions of the heuristic, and/or the collective actions of the files in the set fulfill the conditions of the heuristic.” *Kennedy*, 5:8-11; *see also id.*, 5:12-33 (disclosing numerous heuristics that detect malicious software from related file behavior).

It would have been obvious to configure *Kester's* system/method for monitoring application files such that each remote workstation additionally includes a file set tracking module as disclosed in *Kennedy* to log the set of files for each application in the logging database and each time a file in the set touches (e.g., creates, modifies, and/or reads) another file. *Decl.*, ¶¶109-118. It also would have been obvious to configure *Kester's* application server module and/or database factory to implement *Kennedy's* behavior monitoring module which automatically

determines whether received collection data indicates an application is malware based on an identified relationship between application file(s) and different but related files. *Id.* In the combined method/system, the application's file set behavior is collection data that is sent to the server module and/or database factory such that when the comparison of this collection data to other objects indicates the application is related to known malware, *Kennedy's* automated heuristics can classify the application appropriately as malware. *Id.*

For example, sus-a.exe directly or indirectly creates a new file, sus-b.exe, and attempts to send sus-b.exe to another computer system. *Kennedy*, 4:19-47, 5:12-22. As taught in *Kennedy*, the two application files (sus-a.exe and sus-b.exe) are tracked as part of the same file set and, in the combined method/system, the fact that sus-a.exe created sus-b.exe is recorded by *Kester's* remote workstation(s). *Decl.*, ¶¶109-118. Subsequently, sus-a.exe attempts to send sus-b.exe to another computer system and this event is also recorded. *Id.* This related file behavior is transmitted as part of the collection data to *Kester's* server module and/or database factory. *Id.* When *Kester's* application server module and/or database factory classifies sus-a.exe using the file set behavior monitoring of *Kennedy*, the fact that 1) sus-a.exe created sus-b.exe and 2) sus-a.exe is attempting to send sus-b.exe to another computer system indicate sus-a.exe and sus-b.exe are malicious (i.e., malware). *Id.*

A PHOSITA would have recognized that *Kennedy's* file set tracking and behavior monitoring presents an obvious and predictable improvement to Kester's application file monitoring system/method. *Decl.*, ¶¶109-118. A PHOSITA would have been motivated to apply the teachings of *Kennedy* to enable *Kester's* application file monitoring system/method to include *Kennedy's* file set tracking and behavior monitoring modules to predictably improve the system/method's ability to detect suspicious behavior of different but related files. *Id.* Moreover, *Kennedy's* file set tracking and behavior monitoring would have made *Kester's* method/system more desirable and more effective in identifying malicious/rogue programs (malware), thereby improving similar malware detection method/systems in the same way. *Id.* (describing benefits of automating *Kester's* malware classification process in situations where an application is related to known malware).

Kester recognized a need to monitor and control application files, such as when employees “access server computers to download and execute rogue programs[.]” *Kester*, [0003], [0005]. Indeed, *Kester* already broadly teaches that each remote workstation “monitors the ongoing behavior of the application” even after it determines the application is allowed to run, which a person of ordinary skill would have understood encompasses monitoring the behavior of the set of files associated with the application. *Kester*, [0041]; *Decl.*, ¶¶109-118. *Kester* also

teaches collection data that is logged in the logging database and sent to the base station can “include additional data associated with the application” in addition to the specific examples disclosed. *Kester*, [0041], [0061]. A PHOSITA would have been motivated to apply the teachings of *Kennedy*, which teaches an effective method of tracking and comparing different but related files and their behavior, to identify the types of rogue program threats *Kester* was designed to mitigate more effectively. *Decl.*, ¶¶109-118.

A PHOSITA would have understood this combination of prior art elements would have been accomplished using known methods and with no change to their respective functions. *Id.* For example, *Kester*’s application file monitoring system/method would still allow for monitoring application execution launch and network access behavior while also implementing *Kennedy*’s known method of file set tracking and behavior monitoring to detect any suspicious behavior of related files, which may indicate the presence of a malicious/rogue program (malware). *Id.* Moreover, since file set tracking and behavior monitoring was a well-known way to detect the presence of malware, there would have been a reasonable expectation of success configuring *Kester*’s application file monitoring system/method to perform *Kennedy*’s file set tracking and behavior monitoring. *Id.*

Thus, *Kester* in view of *Kennedy* renders obvious Element 1.5. *Decl.*, ¶¶105-119.

[1.6] classifying, by the base computer, the computer object as malware on the basis of said comparison.

As explained in Element 1.5 above, *Kester* discloses that its application server module 102 and/or application database factory 110 compare merged collection data with “expected network activity” derived “from network activity by a different but related application.” *See* Element 1.5. And *Kester* further discloses this collection data can be ongoingly monitored. *Id.* For the reasons above, it would have been obvious to configure *Kester’s* system/method for monitoring ongoing application file behaviors such that each remote workstation additionally includes a file set tracking module as disclosed in *Kennedy* to log the set of files for each application in the logging database and each time a file in the set touches (e.g., creates, modifies, and/or reads) another file. *Decl.*, ¶¶109-118, 120. In this combination, *Kester’s* application server module and/or database factory (base computer) then utilizes *Kennedy’s* automated heuristics when a relationship between application file(s) and different but related (malicious) file(s) is identified to classify the application as malware. *Id.*, ¶¶120-121. Thus, a PHOSITA would have understood that *Kester’s* modified systems/methods automatically classify an application as malware when the relationship between the application file(s) and different but related files so indicates. *Id.*

Thus, *Kester* in view of *Kennedy* renders Element 1.6 obvious. *Decl.*, ¶¶120-122.

2. Claim 2

2. A method according to claim 1, wherein the relationship is between objects related directly, or between objects related via other objects.

The combined *Kester/Kennedy* systems/methods identify a relationship between correlated data from an application file and other related files. *See* Claim 1, Element 1.5. *Kennedy*'s file set tracking module 312 “tracks the behaviors of the files in the set and records each time a file in the set ‘touches’ another file on the storage device 108 or elsewhere in the computer system 100.” *Kennedy*, 4:19-28 (noting to “‘touch’ means to create, modify, and/or read a file”). A PHOSITA would have understood any files created by the application are “directly” related. *Decl.*, ¶¶123-124.

In addition to direct actions, “‘touching’ can be performed through indirect actions such as launching another process that then touches a file.” *Id.*, 4:31-33. Here, when an application file launches another process that then “touches” another file, there is an indirect relationship between the application file and the touched file. *Decl.*, ¶¶123-124.

3. Claim 3

3. A method according to claim 1, wherein where an object and other objects are found to have a relationship, monitoring those related objects as a homogenous entity and identifying a common object of the homogenous entity as a malware object.

The combined *Kester/Kennedy* systems/methods identify a relationship between correlated data from an application file and other related files. *See* Claim 1, Element 1.5 (describing application of *Kennedy*'s file set tracking functions in *Kester*'s system/method and motivations related thereto). As part of this combined process, *Kennedy* discloses that its file set tracking module 312 tracks the behaviors of a file each time a file “touches” (i.e., creates, modifies, and/or reads another file). *Kennedy*, 4:19-28. Here, the “file set tracking module 312 adds the touched file to the set of the file that touched it.” *Id.*, 4:33-34. A “behavior monitoring module 316” can then “evaluate[] the behaviors of the files...collectively (i.e., tracks the collective actions of the files in the set)” (*Id.*, 4:64-5:3; *see also id.*, claim 6) such that “heuristics” can be run on the collective file set. *Id.*, 5:4-33 (describing several known heuristics).

As explained in more detail in Claim 1, Element 1.5, in an example where *sus-a.exe* directly or indirectly creates a new file, *sus-b.exe*, and attempts to send *sus-b.exe* to another computer system (see *Kennedy*, 4:19-47, 5:13-22), the two application files (*sus-a.exe* and *sus-b.exe*) are tracked as part of the same file set

and when *Kester*'s application server module and/or database factory classifies sus-a.exe in part using the file set behavior monitoring heuristics of *Kennedy*, the fact that 1) sus-a.exe created sus-b.exe and 2) sus-a.exe is attempting to send sus-b.exe to another computer system indicate sus-a.exe and sus-b.exe are malicious (i.e., malware). *See* Claim 1, Element 1.5; *Decl.*, ¶¶109-118, 125-127. A PHOSITA would have understood this process involves identifying a *common object of the homogenous entity* as a malware object. *Decl.*, ¶¶125-127. Namely, each time the application file sus-a.exe creates sus-b.exe across any workstation, the combined *Kester/Kennedy* systems/methods would identify sus-b.exe (common object) as a malware object. *Id.*; *see also* Claim 4 below (discussing how an object can be considered malware based on the behavior of a related object).

4. Claim 4

4. A method according to claim 1, comprising deducing an object to be malware based on the behaviour of a related object.

Kester teaches that “expected network activity can be determined from network activity by a different but related application” so that, for example, “a later version of an application may share common access privileges with an earlier version of the same application.” *Kester*, [0044], [0182]. Although *Kester* does not teach determining an application’s classification category from the behavior of a “related object[,]” such an implementation would have been obvious to a PHOSITA. *See* Claim 1, Element 1.5; *Decl.*, ¶¶109-118, 128.

It would have been obvious to a PHOSITA to configure *Kester's* malware classification process (*see* Claim 1, Elements 1.5-1.6) based on the teachings of *Kennedy's* automation such that if at least one other different but related application file (*related object*) in the same file set as the application file in question (the *object*) performs malicious behavior, the application file is classified as malware. *Decl.*, ¶¶128-129. The combined system/method classifies an application file as malware when at least one related application (i.e., another file in the same file set) performs malicious activity. *Id.* For example, sus-a.exe directly or indirectly creates a new/unknown file, sus-b.exe, and sus-a.exe “sends more than a certain number of emails (e.g., N=10) within a certain time period[,]” indicating malicious activity. *Kennedy*, 4:19-47, 5:23-33. When *Kester's* application server module and/or database factory classifies sus-a.exe using the file set behavior monitoring of *Kennedy*, sus-a.exe is classified as malware because of its email transmission behavior. *Decl.*, ¶¶128-129. The combined method/system classifies sus-b.exe as malware because 1) sus-a.exe created sus-b.exe and 2) sus-a.exe performed malicious behaviors. *Id.*

A PHOSITA would have been motivated and would have appreciated a reasonable expectation of success modifying *Kester* with *Kennedy's* teachings for the reasons detailed above in Claim 1. *See* Claim 1, Element 1.5; *Decl.*, ¶¶109-118. Thus, *Kester* in view of *Kennedy* renders Claim 4 obvious. *Decl.*, ¶¶128-130.

5. Claim 5

5. A method according to claim 1, wherein if at least one other object to which said computer object is related is classed as malware, then classifying said computer object as malware.

See Claim 4 (noting in the combined *Kester/Kennedy* systems/methods, if at least one other different but related application file (other object) in the same file set as the application file in question (the object) performs malicious behavior which would itself be classified as malware, the application file is classified as malware); *Decl.*, ¶131.

6. Claim 6

6. A method according to claim 1, wherein said other objects include the object or similar objects stored on at least one of the first and second remote computers.

Kester teaches the importance of aggregating collection data from multiple remote computers and using the aggregated collection data to make application categorization/classification determinations. See, e.g., *Kester*, [0045] (“In a preferred embodiment, the network activity from multiple workstations is uploaded to the application database factory 110.”), [0119] (the application server module “merges and sorts the collected data including the frequency count with other workstation inventories” and merging and sorting “based on the application or any additional data associated with the application”), [0133] (“the collection data is merged and sorted” at the application database factory at step 1220); see also *id.*,

[0076] (using unexpected behavior to classify/reclassify the application); Claim 1, Elements 1.5, 1.6.

It would have been obvious to configure *Kester's* system/method for monitoring application files such that each remote workstation additionally includes a file set tracking module as disclosed in *Kennedy* to log the set of files for each application in the logging database and each time a file in the set touches (e.g., creates, modifies, and/or reads) another file for the reasons discussed above. *See* Claim 1, Element 1.f. It also would have been obvious to configure *Kester's* application server module and/or database factory to determine whether received data indicates that a program/application is malware based on the behavior and/or classification of different but related files. *See* Claim 1, Elements 1.5 and 1.6; claims 4, 5. Based on *Kester's* teachings to use aggregated collection data from multiple remote workstations to make application classification determinations, in the combined system/method, the different but related files include the application file or similar application files stored on the first or second remote workstations.

Decl., ¶¶132-133

7. Claim 7

7. A method according to claim 1, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said computer object.

Kennedy's file set tracking module 312 “tracks the behaviors of the files in the set and records each time a file [(“parent”)] in the set ‘touches’ [(e.g., creates)] another file [(“child”)] on the storage device 108 or elsewhere in the computer system 100.” *Kennedy*, 4:19-26. According to the ’389 Patent, the “ancestral relationships” of an object to other objects (e.g., “which object created this object, which objects this object created”) are considered when making malware determinations. *See, e.g., ’389 Patent*, 11:5-44; *see also id.*, 17:22-18:15. The object that creates another object is referred to as the “parent” or “father” and the created object is referred to as the “child” or “son[.]” *Id.* Thus, *Kennedy's* file sets include parent and/or child objects and/or otherwise process-related objects. *Decl.*, ¶¶134-135.

For the reasons discussed above, it would have been obvious to implement *Kennedy's* file set tracking and behavior monitoring in the *Kester* system/method. *See* Claim 1, Element 1.5.

8. Claim 8

8. A method according to claim 1, wherein the data about the computer object that is sent from the respective remote computer to the base computer and that is used in the comparison includes one or more of: executable instructions contained within or constituted by the computer object; the size of the computer object; the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object; the creation and modification dates of the computer object; vendor, product and version and any other information stored within the computer object; and the computer object header or header held by the respective remote computer.

Kester teaches the collection data sent from the remote workstations to the base station may include “application name” (*current name of the computer object*), “Original Filename” (*original name of the computer object*), “directory location” (*physical and folder location of the computer object on disk*), “file size” (*size of the computer object*), and “publisher, suite,...version” (*vendor, product and version*). *Kester*, [0061]-[0062], [0051], [0119], [0097], Fig. 6; *see also* Element 1.1; *Decl.*, ¶¶136-137.

9. Claim 9

9. A method according to claim 1, wherein the data is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on the respective remote computer.

Kester teaches the collection data (*see* Claim 1 Element 1.1) may be sent in the form of a hash (key) that is generated by each remote workstation using a hash function. *Kester*, [0051]-[0052] (describing MD-5 and SHA-1 hashing processes); *see also id.*, Claim 39; *Decl.*, ¶¶138-140.

10. Claim 10

10. A method according to claim 9, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.

Kester teaches the “hash for the application is determined by transforming the binary associated with the application into a unique set of bits.” *Kester*, [0052]. “[T]he hash function takes selected binary input from the application and transforms the binary into a fixed-length encrypted output called a hash. The result is a hash with a fixed-size set of bits that serves as a unique ‘digital fingerprint’ for the application.” *Id.* A PHOSITA would have readily understood *Kester*’s hash (key) has at least one component that represents executable instructions contained within the object. *Decl.*, ¶¶141-142. This is consistent with the ’389 Patent’s disclosure of an MD5 Type 1 checksum (aka key). ’389 *Patent*, 10:15-18; *Decl.*, ¶¶141-142.

11. Claim 11

11. A method according to claim 9, wherein the key has at least one component that represents data about said computer object.

Kester's hash value (*key*) includes collection data (*see* Claim 9), which includes “network access data[.]” *Kester*, [0011], Claim 39. “[N]etwork access data includes parsed properties or attributes that are associated with the application when the application accesses the network.” *Kester* at [0069]. Network access data includes, for example, the “time of day, port, I.P. address, protocol” of a network access request, which are events initiated by or involving the object when the object runs on the respective remote computers. *Kester*, [0061], [0051], [0011], [0083]; *Decl.*, ¶143. Additionally, or alternatively, network access data includes the “name, publisher, suite, [] file size, [and/or] version” of the network accessing application. *Kester*, [0051]; *Decl.*, ¶143. Thus, *Kester's* hash value has at least one component that represents data about the object. *Decl.*, ¶¶143-144

12. Claim 12

12. A method according to claim 11, wherein said data about said computer object includes at least one of: the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object; the creation and modification dates of the computer object; vendor, product and version and any other information stored within the computer object; the computer object header or header held by the respective remote computer; and, events initiated by or involving the computer object when the computer object is created, configured or runs on the respective remote computers.

See claim 8.

13. Claim 13

13. A method according to claim 9, wherein the key has at least one component that represents the physical size of the computer object.

Kester's hash value (*key*) includes collection data (see Claim 7), which includes the network accessing application's "file size[.]" *Kester*, [0051], [0011], Claim 39. Thus, *Kester's* hash value has at least one component that represents the physical size of the object. *Decl.*, ¶¶146-147; see also analysis for Claims 8, 9.

14. Claim 15

15[P]. An apparatus for classifying a computer object as malware, the apparatus comprising:

To the extent the preamble is found limiting, *Kester* teaches an apparatus for classifying a computer object as malware. See Claim 1, Element 1.6 above.

[15.1] a base computer constructed and arranged to receive 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;

See Claim 1, Element 1.1.

[15.2] the base computer being constructed and arranged to receive data about the computer object from a second remote computer on which the computer object or similar computer objects re [sic] 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;

See Claim 1, Element 1.2.

[15.3] the base computer being constructed and arranged to correlate 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;

See Claim 1, Element 1.4.

[15.4] the base computer being constructed and arranged to compare the 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

See Claim 1, Element 1.5.

[15.5] the base computer being constructed and arranged to classify the computer object as malware on the basis of said comparison.

See Claim 1, Element 1.6.

15. Claim 16

16. Tha [sic] apparatus according to claim 15, wherein the base computer is constructed and arranged so as to identify the relationship between objects related directly, or between objects related via other objects.

See Claim 2.

16. Claim 17

17. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as, where an object and other objects are found to have a relationship, to monitor those related objects as a homogenous entity and identify a common object of the homogenous entity as a malware object.

See Claim 3.

17. Claim 18

18. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to deduce an object to be malware based on the behaviour of a related object.

See Claim 4.

18. Claim 19

19. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that if at least one other object to which said object is related is classed as malware, then said object is classified as malware.

See Claim 5.

19. Claim 20

20. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include the object or similar objects stored on at least one of the first and second remote computers.

See Claim 6.

20. Claim 21

21. The apparatus according to claim 15, wherein the base computer is constructed and arranged so that said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

See Claim 7.

21.Claim 22

22. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to compare the data where the data includes one or more of: executable instructions contained within or constituted by the computer object; the size of the computer object; the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer object; the creation and modification dates of the computer object; vendor, product, and version and any other information stored within the computer object; and the computer object header or header held by the respective remote computer.

See Claim 8.

22.Claim 23

23. The apparatus according to claim 15, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of a key that is obtained by a hashing process carried out in respect of the objects on said respective remote computer.

See Claim 9.

23. Claim 24

The apparatus according to claim 23, wherein the key has at least one component that represents executable instructions contained within or constituted by the computer object.

See Claim 10.

24. Claim 25

25. The apparatus according to claim 23, wherein the key has at least one component that represents data about said computer object.

See Claim 11.

25. Claim 26

26. Apparatus according to claim 25, wherein said data about said object includes at least one of the current name of the computer object; the physical and folder location of the computer object on disk; the original name of the computer [sic] object; the creation and modification dates of the computer object; vendor, product, and version and any other information stored within the computer object; the computer object header or header held by the respective remote computer, and events initiated by or involving the computer object when the computer object is created, configured, or runs on the respective remote computers.

See Claim 12.

25. Claim 27

27. The apparatus according to claim 23, wherein the key has at least one component that represents the physical size of the computer object.

See Claim 13.

26. Claim 29

29. A computer program recorded on non-transitory computer readable medium comprising program instructions for causing a computer to perform the method of claim 1.

Kester teaches a non-transitory computer readable medium (e.g., “addressable storage medium”) comprising program instructions for causing a computer to perform the method steps of claim 1. *Kester*, [0034] (“[M]any of the components of the various systems which may be included in the entire system, some of which are referred to as modules, can be implemented as software, firmware or a hardware component, such as a field programmable gate array (FPGA) or application specific integrated circuit (ASIC), which performs certain tasks. Such components or modules may be advantageously configured to reside on the addressable storage medium and configured to execute on one or more processors.”); *see also* Claim 1; *Decl.*, ¶166.

27. Claim 30

30. A method according to claim 1, wherein the data received about the computer object from the first remote computer is different than the data received about the computer object from the second remote computer.

Kester discloses receiving “collection data” about a computer object from a first and second remote computer. *See* Claim 1, Elements 1.1 and 1.2 above. This “collection data” includes information about events involving the application when the application runs on the respective remote workstation. *Id.* For example, “collection data” includes the “name, publisher, suite, hash, file size, version, protocol, I.P. address, port, and additional information or properties which are associated with a launched or network accessing application.” *Kester*, [0051], [0061] (further collecting “request frequency” and “time of execution”), [0073] (collecting “trace ID”), [0083], Fig. 4B; *see also* Claim 8 above. *Kester* also generally contemplates collecting “other network access data” (*see Kester*, [0073]) such that “the collection data can include any additional data associated with the applications[.]” *Id.*, [0108]; *see also* Claim 1, Element 1.1 (describing additionally collecting “event type”).

A PHOSITA would have understood that *Kester’s* systems/methods encompass a scenario where the collection data received from the first workstation is different from the collection data received from the second workstation. *Decl.*,

¶¶167-168. For example, when workstations 1 and 2 upload collection data related to the application to the application server module and/or application database factory, a PHOSITA would have understood the collection data from this application launching or requesting network access from different workstations would include: different versions; file sizes; protocols; requests to access different destination IP addresses and different ports; different request frequencies or times of execution; and any other “different” collection data (e.g., different directory location, publisher, suite, name, etc.). *Id.*

B. Ground 2: Kester in view of *Kennedy* and *Honig* Renders Claims 14 and 28 Obvious

Ground 2 is evaluated under the same standards as Ground 1 summarized above.

1. Claim 14

14. A method according to claim 1, comprising initially classifying a computer object as not malware, 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.

Petitioner interprets claim 14’s method steps to permit a process that operates on a “computer object” that is different from the computer object of claim 1 (from which claim 14 depends). First, the claim language does not suggest that

claim 14's "computer object" is antecedently based on claim 1's "computer object." Claim 14 recites "a computer object," rather than "the computer object"—the latter of which would be proper were the patentee intending to antecedently connect the "computer objects" in these claims.

Second, the functionality recited in these claims suggests different "computer objects" may be analyzed in claim 1 and in claim 14. Indeed, the final method step in claim 1 requires "classifying . . . the computer object as malware on the basis of" a comparison step that identifies a relationship, e.g., that the computer object is related to known malware. So, to practice all steps of claim 1, an analysis of the "computer object" identifies a relationship and ends with a determination that said object is malware. Claim 14 begins by "initially classifying a computer object as not malware" and ends by "reclassifying the computer object as malware" if the object's behavior exceeds that permitted by a mask. In other words, claims 1 and 14 recite two methods of classifying computer objects as malware that do not necessarily operate together or apply to the same object. Because claim 1 ends with a computer object being classified as malware and claim 14 begins by "initially classifying a computer object as not malware," the claim 14 steps may relate to a different object than the claim 1 steps—claim 1's computer object will already have been classified as malware and so a different object would then be *initially* classified as not malware. Even if claim 14's steps

were performed before claim 1's steps (or within the claim 1 method), there is no apparent way to complete every step of claim 14 and every step of claim 1 with a single computer object. Each distinct process ends by either classifying or reclassifying a computer object as malware. So, completing claim 1's relationship-based classification obviates the need for claim 14's mask-based classification/reclassification. Similarly, completing claim 14's mask-based classification/reclassification obviates the need for claim 1's relationship-based classification. Accordingly, to give meaning to each recited method step in claims 1 and 14, the latter should be interpreted to capture a process that analyzes a different computer object than that analyzed in claim 1.

Further, claims in a related patent expressly state that only one of these two classification methods will be used to classify a given computer object as malware. Namely, independent claim 1 and dependent claims 3-4 from related U.S. Patent No. 8,418,250 ("the '250 Patent") similarly recite relationship-based and mask-based classification processes, but expressly present these processes as alternatives. Specifically, the first three limitations from '250 Patent claim 1 recite classifying a computer object as malware when data about the computer object indicates the object is malware. EX1021 ('250 Patent), Claim 1. Dependent claims 3-4 further specify that the data indicates that the object is related to known malware. *Id.* at Claims 3-4. The remaining limitations of '250 Patent claim 1 recite a mask-based

process, similar to '389 Patent claim 14, in which the object is initially classified as not malware, a mask is generated, and the object is reclassifying as malware if its behavior exceeds the mask. *Id.* at Claim 1. Importantly, '250 Patent claim 1 expressly states that the mask-based process proceeds only if the relationship-based process concludes the object is not malware. *Id.* Accordingly, if the relationship-based process results in classifying an object as malware, the '250 Patent claim 1 method never proceeds to the mask-based process. Similarly, if the mask-based process is invoked in '250 Patent claim 1, it means the relationship-based process *did not* classify the object as malware.

Accordingly, Petitioner interprets claim 14 to read on a process that is separate from the method steps of claim 1 and operates on a distinct “computer object.” This interpretation is consistent both with the express language of the claims and with claims directed to similar subject matter in a related patent.

i. initially classifying a computer object as not malware

Kester in view of *Kennedy* and *Honig* teaches a process for initially classifying an application not as malware. As set forth in detail below, Ground 2 proposes that a PHOSITA would have been motivated pursuant to *Honig* to automate certain classification functionality in *Kester*'s system. Namely, *Kester* describes a process by which a human administrator classifies application files based on certain collected data, and Ground 2 proposes that a PHOSITA would

have been motivated to automate this process pursuant to Honig's machine learning teachings. For clarity, Ground 2 does not advance a theory that the Challenged Claims are satisfied through any processes that require human involvement.

As explained in claims 1.1 and 1.2 above, *Kester's* application server module and/or application database factory (base computer) receives "collection data" from each workstation, where such collection data can be uploaded from multiple (potentially thousands) of workstations. *See above* Claims 1.1 and 1.2 (*citing Kester*, [0077], [0079], [0095], [0119], [0162]). These workstations can be running the same application (*Kester*, [0157]) or different applications. *Kester*, [0077], [0079], [0095], [0119], [0162]. Petitioner's analysis herein is directed to *Kester's/Honig's* modified functionalities with respect to a different application than the application subject to *Kester's/Kennedy's* combined functionalities in claim 1.

Kester discloses that each application file can be associated with one or more parent groups and/or categories. An exemplary parent group is "malware." *Kester*, [0015], [0081]-[0082], [0098]. An exemplary application category is "malicious applets and scripts"—i.e., malware. *See '389 Patent*, 1:19-22 (defining "malware"); *Decl.*, ¶¶170-172; *see above* Element 1.5 (annotated Fig. 4).

To determine which category to assign an application, Kester derives expected network access data from applications behaving in a predetermined manner; the application server module 102 monitors “the behavior of categorized and uncategorized applications[;]” and in an automated step, a “categorized application that does not operate in a predetermined manner is placed in the uncategorized applications database 108.” *Kester*, [0070]. A network administrator may then interface with the application server module 102 to “classify uncategorized applications and/or recategorize previously categorized applications[.]” *Kester*, [0134]; *see also id.*, [0071], [0078], [0084], [0122], Figs. 3, 10; *see also id.*, Fig. 20.

Because *Kester* teaches assigning an appropriate category and parent group, such as “malicious applets and scripts[,]” and “malware” for an application based on behavior and additional data associated with the application, a PHOSITA would have understood that Kester classifies an application, e.g., as malware or not. Decl., ¶¶170-173.

For example, an exemplary user display is shown below:

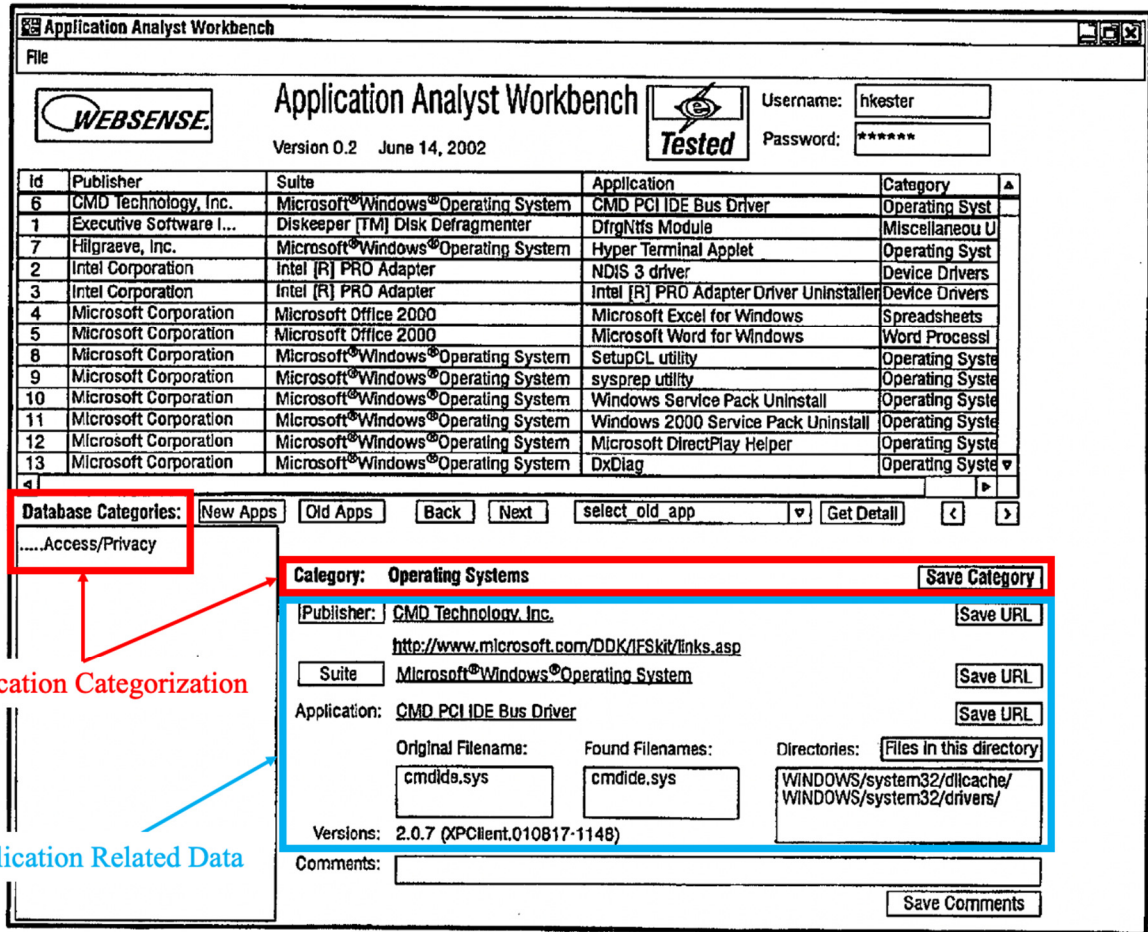


FIG. 6

Kester, Fig. 6 (annotated).

The application related data for the “CMD PCI IDE Bus Driver” application in Figure 6 above includes the publisher (CMD Technology, Inc.), related suite (Microsoft Windows Operating System), filename, and directory location, indicating the application is not malware. *Kester*, [0097]; Fig. 6; *Decl.*, ¶173. Accordingly, “the application analyst’s classification module 302 classified the application in the parent group titled ‘access/privacy’” and “Operating System” category (i.e., not malware). *Kester*, [0098], Fig. 6; *see also Kester*, Claim 19.

While *Kester's* process initially classifies an application, including as not malware (e.g., parent group titled “access/privacy”), it does so with human involvement. It would have been obvious to automate *Kester's* application categorization process, such that the application server module and/or database factory uses an adaptive learning system (rather than or in addition to a human reviewer) to determine whether received data indicates the application is malware, as taught by *Honig. Decl.*, ¶¶174-180.

Honig teaches various adaptive learning systems that determine whether data is malware. *Honig* discloses a model generator 16 that automatically generates a model of behavior for a computer object. *Honig*, 8:15-18 (“**The data in the data warehouse 14 is accessed by the detection model generators 16 which generate models that classify activity as either malicious or normal.**”), 24:43-45; *see also id.*, 8:4-24 (noting *Honig's* framework can “handle virtually any data type” as input, including “audit data”). *Honig* teaches three exemplary types of model generation algorithms supported by the system: “The first is misuse detection, which trains on labeled normal and attack data. The second is supervised (traditional) anomaly detection which trains on normal data. The third is unsupervised anomaly detection which trains on unlabeled data.” *Honig*, 20:33-21:38.

These model generation algorithms were well known prior to 2005 and *Honig* discloses specific example implementations of each. *Honig*, 2:5-4:47, 21:39-24:32; *Decl.*, ¶176. *Honig* teaches each such model generation algorithm is “fully automated.” *Honig*, 7:17-20 (“Automated model generation trains detection models from data stored in the data warehouse. The process for converting the data into the appropriate form for training is fully automated”); *see also id.*, 20:33-21:38. A PHOSITA would have understood *Honig* to disclose an adaptive learning system that programmatically determines whether data is malware. *Honig*, 7:38-41; *Decl.*, ¶176.

It would have been obvious to modify *Kester*’s application server module and/or database factory to initially determine whether received data indicates a program/application is malware as taught by *Honig*. *Decl.*, ¶¶174-180. In the combined *Kester/Honig* method/system, *Kester*’s server module and/or database factory would categorize applications using an adaptive learning model, as *Kester* already contemplates. *See Kester*, [0092] (“Categorization can be based upon word analysis, **adaptive learning systems**, and image analysis”). Such an implementation would not preclude a human reviewer from separately validating and/or modifying the automated determination, and therefore would not fundamentally change the overall operation of *Kester*’s method. *Decl.*, ¶¶174-180. A PHOSITA would have been motivated to implement the combined *Kester/Honig* method in this way to

reduce or eliminate the workload of the human reviewer, furthering one of the expressly stated goals of Kester—to “enhance productivity of the human reviewer.” *Kester*, [0090]; *Decl.*, ¶¶174-180.

More broadly, a PHOSITA would have been motivated to configure *Kester*’s server module and/or database factory to use an adaptive learning model to initially determine whether received collection data indicates the application is malware as taught by *Honig* because this combination would have been a combination of prior art elements according to known methods to yield predictable results. *Id.* As demonstrated by *Honig*, automating malware classification determinations through adaptive learning was known in the prior art. *Id.* A PHOSITA would have understood such a modification would have been well within the capabilities of one of ordinary skill in the art such that there would have been a reasonable expectation of success. *Id.*

- ii. *generating a mask for said computer object that defines acceptable behaviour for the computer object*

With respect to the claimed “mask,” in litigation, PO interprets the plain meaning of these “mask” limitations to “encompass[] any suitable way to define acceptable behavior” for an object. EX1007 (Dkt. 98, *PO Opposition Brief*), 12; *see also id.*, 11 (“In other words, the mask establishes the permissible and impermissible behavior of an object”). Petitioner asks the Board apply PO’s

interpretation of the plain meaning of this limitation for the purposes of this proceeding. *Supra*, Section II.D.

As explained in Claim 1 above, *Kester* teaches its application categorization and classification process includes determining an application's "**expected or allowed network activity**" (i.e., *generating a mask*). See *Kester*, [0179]-[0181], Fig. 20; see also *id.*, [0070] ("expected network access data is derived from network access data obtained when the application is behaving in a predetermined manner"). Namely, *Kester* teaches a human reviewer (at the application server module 102 and/or application database factory 110) interacts with a classification user interface to determine the "expected or allowed network activity" (also referred to as expected or predetermined behavior) for an application from, for example, a record of the "application's prior activity" or contemporaneous activity on one or multiple workstations. *Kester*, [0179]-[0181]; *Kester*, [0183] ("the network activity from multiple workstations is uploaded to the application database factory 110"); see generally *Kester*, [0178]-[0182], [0043]-[0044], [0069]-[0070], [0163]-[0164], Figs. 17, 20.

Kester's expected network activity includes one or more network attributes (also referred to as properties or features) that are associated with the application when the application accesses the network in an expected manner. The attributes can include a specific protocol, IP address and/or port. *Kester*, [0046], [0184]; see

also Kester, [0070] (“expected features relate to normal network activity by the application”); *Kester*, [0083] (“The network access data [including protocol, IP address, and/or port] for the application allows the network administrator to discriminate between expected/predetermined behavior and unexpected behavior of the application.”); *Kester*, Fig. 4B.

Like *Kester*, the ’389 Patent discloses a mask that “can be built up with use of that object[,]” “can give an overall picture of the expected ‘normal’ behaviour of that object[,]” and might indicate “details of any ports or interfaces” for legitimate Internet connections. ’389 *Patent*, 14:61-15:10. Accordingly, a PHOSITA would have understood *Kester* teaches generating a mask for the computer object that defines acceptable behavior for the object. *Decl.*, ¶¶181-195.

While *Kester*’s mask generation involves human input, *Kester* explains the process for classifying or categorizing applications “can be based upon [] adaptive learning systems[.]” *Kester*, [0089], [0092], Fig. 20. A PHOSITA would have found it obvious to look to known adaptive learning systems to perform *Kester*’s classification and/or categorization processes. *Decl.*, ¶¶181-195. As described above, *Honig* teaches a known process for generating adaptive learning models to inform malware classifications. *See Claim 14.i.*

As part of *Honig*’s techniques, *Honig* discloses a model generator 16 that automatically generates a model of normal behavior for a computer object. *Honig*,

24:43-45 (“The model generator 16 then uses the unsupervised SVM algorithm to generate a model of normal activity”); *see also id.*, 4:54-57 (“What is needed is an architecture to automate the processes of data collection, model generation and data analysis, and to solve many of the practical problems associated with the deployment of data mining-based IDSs.”). In *Honig*, the model generation algorithms are “fully automated.” *Honig*, 7:17-20; *see also id.*, 20:33-21:38. This model of normal behavior can then be used to make a malware classification. *Decl.*, ¶¶181-195.

It would have been obvious to automatically generate (e.g., without human input) Kester’s mask in view of the teachings of *Honig*. *Decl.*, ¶¶181-195. A PHOSITA would have been motivated to generate *Kester*’s “expected or allowed network activity” (mask) for an application using an adaptive learning model as disclosed in *Honig* because it is expressly contemplated by *Kester*. *Kester*, [0089], [0092], Fig. 20 [0090]; *Decl.*, ¶¶181-195. A PHOSITA would have also recognized that *Honig* provides an express benefit for implementing the combined method/system in this way—to allow the method/system to “update and alter models on the fly[,]” consistent with *Kester*’s framework for allowing an administrator to adjust/control prohibited behaviors. *Id.*; *Honig*, 19:46-62.

A PHOSITA would have recognized automating mask generation to be an obvious and predictable improvement to *Kester*’s server module and/or database

factory classification software, given that *Kester* expressly contemplates use of adaptive learning systems. *Decl.*, ¶¶181-195. Moreover, *Honig* recognized the need for an extensible and scalable system where the model generator components are modular components that are “plugged” into the system architecture. *Honig*, 7:3841, 13:61-66. *Honig* provides a detailed discussion of adaptive learning model generators that were known in the field as of 2005 and provides a detailed discussion of example implementations. *Honig*, 2:5-4:47, 21:39-24:32; *Decl.*, ¶¶181-195. Adaptive learning model generation dates back to at least 1993, when the technique was used at SRI in the Emerald system. *Honig*, 3:31-41; *Javitz et al*, “*The NIDES Statistical Component: Description and Justification*” (EX1019); *Decl.*, ¶193. As a result, this modification would have been well within the capabilities of a PHOSITA such that there would have been a reasonable expectation of success. *Decl.*, ¶¶181-195. Moreover, automating generation of a mask of expected/allowed behavior, as taught by *Honig*, would have made *Kester*’s method/system more desirable and more efficient for a human reviewer to use to confirm results, thereby improving similar malware detection method/systems in the same way. *Id.*; *Kester*, [0090], [0135] (“hints” enhance productivity of human reviewer).

Thus, *Kester* in view of *Honig* renders Element 14.ii obvious.

- iii. *monitoring an operation of the computer object on at least one of the first and second remote computers*

Kester teaches its network access detection module 208 on each remote workstation continually monitors the operation of the application (*the object*):

[T]he execution launch detection module 210 initially evaluates a launched application and allows the application to run[.] ...

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

Kester, [0137]-[0138]; *see also id.*, [0041] (“[T]he workstation management module 200 can monitor the ongoing network activity or behavior of the application”), [0051] (“the network access detection module 108 monitors the behavior or activity of the application or program”); [0139] (“FIG. 14 is a flow diagram illustrating a process for monitoring the behavior of an application.”); *see generally id.*, [0137]-[0146], [0039], [0050]-[0055], [0057]-[0060], Figs. 2, 14.

- iv. *and reclassifying the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.*

Kester teaches that if an application is “not operating in a predetermined manner[.]” it is “categorized or re-categorized[.]” *Kester*, [0076]. Re-categorization can occur at the application server module 102 or application factory database 110.

Namely, when an application is not operating in a predetermined/expected manner (*the actual monitored behavior extends beyond that permitted by the mask*), the application is considered “uncategorized” and the collection data is uploaded from the workstation(s) to the application server module. *Kester*, [0074]-[0075]; *see also id.*, [0070], [0084], [0151], [0153]. The network administrator can then “recategorize” the application based on the comparison of the collection data with the predetermined/expected behavioral data. *Kester*, [0071]; *Decl.*, ¶¶197-200.

Regarding factory 110, an uncategorized application’s collection data may be uploaded to the application database factory for reclassification/re-categorization, for example, “immediate[ly]” (e.g., upon receipt from the workstation(s)) (*see Kester*, [0078]) or when the human reviewer “does not classify the application” (*see Kester*, [0076]). *See also id.*, [0078], [0084], [0122], Fig. 3. At the application database factory 110, “[t]he application analyst classification module 302 can use SQL queries, in conjunction with the rules, to select the subset [of applications] for categorization **or recategorization** from the master application database 300.” *Kester*, [0091]. The reclassification/recategorization process in the application database factory is the same as the classification/categorization process discussed previously. *Kester*, [0089]-[0094]; *Decl.*, ¶¶197-200. *Kester* therefore teaches reclassifying the program/application as malware when the collection data so indicates. *Id.*

For the reasons above, it would have been obvious to implement *Kester's* application classification process, such that the application server module and/or database factory classify applications into the malware parent group/category without human intervention using an adaptive learning model as taught by *Honig*. See Claim 14.i; *Decl.*, ¶¶174-180, 197-200. For those same reasons, it would have been obvious, and a PHOSITA would have appreciated a reasonable expectation of success, implementing *Kester's* application classification process, such that the application server module and/or database factory re-classify applications into the malware parent group/category using *Honig's* adaptive learning model. *Decl.*, ¶¶174-180, 197-200

2. Claim 28

14. The apparatus according to claim 15, wherein the base computer is constructed and arranged to generate a mask for a computer object that is initially classed not as malware, said mask defining acceptable behaviour for the computer object, and the base computer is constructed and arranged to monitor an operation of the computer object on at least one of the first and second remote computers and to reclassify the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.

See claim 14.

VI. THE FINTIV AND GENERAL PLASTIC FACTORS FAVOR INSTITUTION

Under the “*Fintiv* factors,” the Board may consider parallel litigation, including an early trial date, in determining whether to institute under 35 U.S.C. § 314(a). *NHK Spring Co., Ltd., v. Intri-Plex Technologies, Inc.*, IPR2018-00752, Paper 8 at 19–20 (PTAB Sept. 12, 2018) (precedential); *Apple Inc. v. Fintiv, Inc.*, IPR2020-00019, Paper 11 at 2–3, 6 (PTAB Mar. 20, 2020) (precedential). Those factors favor institution here.

A. Stay

Parallel litigation is ongoing. While no stay has been requested, Petitioner may seek a stay if institution is granted, thus favoring institution. At worst, this factor is neutral because the Board “will not attempt to predict” how the district court will proceed. *Sand Revolution II, LLC v. Continental Intermodal Group-Trucking LLC*, IPR2019-01393, Paper 24 at 7 (PTAB Jun. 16, 2020).

B. Proximity of the Court’s Trial Date

“This factor looks at the proximity of the trial date to the date of [FWD] to assess the weight to be accorded a trial date set earlier than the expected [FWD] date.” *Shenzhen Carku Tech. Co., Ltd. v. The Noco Co.*, IPR2020-00944, Paper 20 at 61 (PTAB Nov. 12, 2020). The District Court’s current scheduling order sets

August 19, 2024 as the trial date for each defendant in five separate lawsuits consolidated only for pretrial issues. Ex. 1009 (Fifth Amended Scheduling Order), 6. Because the court's consolidation is limited to pretrial issues, separate trials will be required, and trial cannot proceed on this date for all defendants, so the trial for Trend Micro may occur later. The Board's FWD in this proceeding will be due in September 2024, likely around when trial against Petitioner or any other defendant commences. Accordingly, an FWD is expected at about the same time as trial. This factor weighs heavily in favor of institution.

C. Investment in Parallel Proceeding

Beyond claim construction, the parties have invested little in the parallel proceeding. Webroot has served initial infringement contentions covering all asserted patents and Trend Micro served invalidity contentions covering the first six patents (including the '389 Patent). Fact discovery does not close until January 18, 2024, and dispositive motions are not due until May 7, 2024. EX1009 (Fifth Amended Scheduling Order), 6. This factor weighs in favor of institution.

D. Overlap

Petitioner hereby stipulates that if institution of this IPR is granted Petitioner will not raise in the parallel District Court proceedings any ground raised in the

IPR. *Sand Revolution II, LLC*, IPR2019-01393, Paper 24 at 11. Accordingly, this factor strongly supports institution.

E. Same Party

The Parties are the same. However, members of the Board have noted *Fintiv* addresses only where the petitioner is unrelated to a defendant in a parallel proceeding, finding this should weigh against denying institution, but *Fintiv* “says nothing about situations in which the petitioner is the same as, or is related to, the district court defendant.” *Cisco Sys., Inc. v. Ramot at Tel Aviv Univ. Ltd.*, IPR202000122, Paper 15 at 10-11 (PTAB May 15, 2020) (APJ Crumbley, dissenting) (disfavoring a “defendant in the district court” is “contrary to the goal of providing district court litigants an alternative venue to resolve questions of patentability”).

F. Other Circumstances

“[W]here the PTAB determines that the information presented at the institution stage presents a compelling unpatentability challenge, that determination alone demonstrates that the PTAB should not discretionarily deny institution under *Fintiv*.” *Hewlett Packard Enterprise Co., et al. v. Billjco LLC*, IPR2022-00420, Paper 17, at 6 (PTAB July 12, 2022) (“[c]ompelling, meritorious challenges are those in which the evidence, if unrebutted in trial, would plainly

lead to a conclusion that one or more claims are unpatentable by a preponderance of the evidence.”).

G. General Plastic Factors

None of the factors articulated in *General Plastic Industrial Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 16 (PTAB Sept. 6, 2017) weigh in favor of discretionary denial. Where a petitioner files a “copycat” petition as its first petition in conjunction with a timely motion for joinder and agrees to assume a “passive understudy role,” the *General Plastic* factors are “effectively neutraliz[ed].” *Apple Inc. v. Uniloc 2017 LLC*, IPR2018-00580, Paper 13 at 10-11 (Aug. 21, 2018).

VII. CONCLUSION

For the foregoing reasons, Petitioner respectfully requests *inter partes* review of the Challenged Claims.

Respectfully submitted,
The Marbury Law Group, PLLC

A handwritten signature in black ink, reading "Robert M. Hansen", is written over a horizontal line.

Robert M. Hansen, Reg. No. 43,656
Matthew J. Anderson, Reg. No. 78,257
Scott B. Amankwatia, Reg. No. 78,226

Attorneys for Petitioner

Date: October 4, 2023

The Marbury Law Group, PLLC
11800 Sunrise Valley Drive
15th Floor
Reston, VA 20191
Tel: (703) 391-2900

VIII. Mandatory Notices (37 C.F.R. § 42.8(a)(1))

REAL PARTY IN INTEREST:

The real parties-in-interest are Trend Micro, Inc., Trend Micro America, Inc. and Trend Micro, Incorporated. Trend Micro, Inc. is a wholly owned subsidiary of Trend Micro America, Inc., which is a wholly owned subsidiary of Trend Micro Incorporated, a Japanese Corporation.

RELATED MATTERS:

Pursuant to 37 C.F.R. § 42.8(b)(2), Petitioner identifies the following related matters.

A petition for *inter partes* review was filed with respect to U.S. Patent No. 8,726,389 by Crowd Strike as IPR2023-00556. This IPR has been instituted and joinder is respectfully requested based on this Petition.

A petition for *inter partes* review IPR2023-00655 was filed by Petitioner with respect to U.S. Patent No. 8,418,250, which is the parent of the '389 patent.

The '389 patent is also at issue in the following district court litigations:

- *Webroot, Inc. and Open Text, Inc. v. Trend Micro Inc.*, Case No. 6:22-cv-00239 (W.D. Tex.), which was originally filed on March 4, 2022.
- *Webroot, Inc. and Open Text, Inc. v. Sophos LTD.*, Case No. 6:22-cv-00240 (W.D. Tex.), which was originally filed on March 4, 2022.

- *Webroot, Inc. and Open Text, Inc. v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.*, Case No. 6:22-cv-00241 (W.D. Tex.), which was originally filed on March 4, 2022.
- *Webroot, Inc. and Open Text, Inc. v. AO Kaspersky Lab*, Case No. 6:22-cv-00243 (W.D. Tex.), which was originally filed on March 4, 2022.
- *Webroot, Inc. et al. v. Forcepoint LLC*, Case No. 6:22-cv-00342 (W.D. Tex.), which was originally filed on March 4, 2022.

LEAD AND BACKUP COUNSEL:

Pursuant to 37 C.F.R. § 42.8(b)(3) and 37 C.F.R. § 42.10(a), Petitioner appoints:

Robert M. Hansen (Reg. No. 43,656) as lead counsel,

Scott B. Amankwatia (Reg. No. 78,226) as back-up counsel, and

Matthew J. Anderson (Reg. No. 78,257) as back-up counsel.

Pursuant to 37 C.F.R. § 42.10(b), a power of attorney is being filed with this designation of counsel.

SERVICE INFORMATION:

Petitioner provides the following service information for designated counsel:

The Marbury Law Group, PLLC
11800 Sunrise Valley Drive
15th Floor
Reston, VA 20191
Tel: (703)-391-2900
Fax: (703)-391-2900

Petitioner consents to electronic service by email to the email addresses:

rhansen@marburylaw.com;

samankwatia@marburylaw.com;

manderson@marburylaw.com; and

pat-docketing@marburylaw.com.

PAYMENT OF FEES

The PTO is authorized to charge any fees due during this proceeding to
Deposit Account No. 18-1579.

EXHIBIT LIST

Exhibit No.	Description
1001	U.S. Patent No. 8,726,389 to Morris et al. (“389 Patent”)
1002	File History of U.S. Patent No. 8,726,389 (“389 File History”)
1003	Declaration of Dr. Wenke Lee (“Decl.”)
1004	U.S. Patent Application Publication No. 2005/0210035 A1 to Kester et al. (“Kester”)
1005	U.S. Patent No. 7,594,272 to Kennedy et al. (“Kennedy”)
1006	U.S. Patent No. 7,225,343 to Honig et al. (“Honig”)
1007	<i>PO Opposition Markman Brief</i> , 22-cv-00243 WDTX, No. 98
1008	<i>Defendants’ Markman Brief</i> , 22-cv-00243 WDTX, No. 86
1009	Order Granting Fifth Amended Scheduling Order, 22-cv-00243 WDTX, No. 252 (“ <i>Fifth Amended Scheduling Order</i> ”)
1010	SANS Institute (2000-2005) <i>Host- vs. Network-Based Intrusion Detection Systems</i>
1011	Intrusion Detection with Unlabeled Data Using Clustering by Leonid Portnoy, et al. (“ <i>Portnoy</i> ”)
1012	U.S. Patent No. 6,944,772 to Dozortsev (“ <i>Dozortsev</i> ”)
1013	U.S. Patent No. 6,772,346 to Chess et al. (“ <i>Chess</i> ”)
1014	WO 2002/033525 to Shyne-Song Chuang (“ <i>Chuang</i> ”)
1015	EP 1,549,012 to Kristof De Spiegeleer (“ <i>De Spiegeleer</i> ”)
1016	EP 1,280,040 to Alexander James Hinchliffe, et al. (“ <i>Hinchliffe</i> ”)
1017	U.S. Patent No. 7,089,428 to Farley et al. (“ <i>Farley</i> ”)
1018	U.S. Patent Publication No. 2004/0153644 to McCorkendale (“ <i>McCorkendale</i> ”)
1019	The NIDES Statistical Component: Description and Justification by Harold S. Javitz et al.
1020	U.S. Patent No. 7,516,476 to Kraemer et al. (“ <i>Kraemer</i> ”)
1021	U.S. Patent No. 8,418,250 to Morris et al. (“250 Patent”)
1022	U.S. Patent No. 10,284,591 to Giuliani et al. (“591 Patent”)

CERTIFICATION OF SERVICE (37 C.F.R. §§ 42.6(e), 42.105(a))

The undersigned hereby certifies that the above-captioned **PETITION FOR INTER PARTES REVIEW OF U.S. PATENT NO. 8,726,389**, Petitioner's power of attorney, and all associated exhibits were served in their entireties on October 4, 2023, on the following party via Federal Express to the Patent Owner by serving the correspondence address of record for the '389 patent:

**Merchant & Gould P.C.
150 South Fifth Street
Suite 2200
Minneapolis, MN 55402:**

A handwritten signature in black ink, reading "Robert M. Hansen", written over a horizontal line.

Robert M. Hansen, Reg. No. 43,656
Attorney for Petitioner

Date: October 4, 2023

CERTIFICATE OF COMPLIANCE WITH WORD COUNT LIMITATION

Pursuant to 37 C.F.R. § 42.24 *et seq.*, the undersigned certifies that this Petition complies with the 14,000-word type-volume limitation. This Petition contains **13,962** words according to the word-processing program used to prepare this document (Microsoft Word), excluding words in figures copied from the prior art, tables of contents and authorities, mandatory notices under 37 C.F.R. §42.8, certificates of service or this certificate of word count, and appendix of exhibits.

THE MARBURY LAW GROUP, ^{PLLC}

A handwritten signature in black ink, reading "Robert M. Hansen", with a horizontal line drawn underneath the signature.

Robert M. Hansen, Reg. 43,656
Attorney for Petitioner

Date: October 4, 2023

The Marbury Law Group, PLLC
11800 Sunrise Valley Drive
15th Floor
Reston, VA 20191
Tel: 703-391-2900