

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AO KASPERKY LAB,

Petitioner,

v.

OPEN TEXT INC.,

Patent Owner.

IPR2023-00556
Patent No. 8,726,389

PATENT OWNER'S RESPONSE

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

TABLE OF CONTENTS

	Page(s)
I. INTRODUCTION	1
II. BACKGROUND	2
A. The State of the Art.	2
B. The '389 Patent	3
C. Claim Construction	9
D. The Level of Ordinary Skill	10
III. OVERVIEW GROUNDS AND CITED ART.	11
A. Kester.....	11
B. Kennedy.....	13
C. Honig	15
IV. LEGAL STANDARD	17
V. GROUND 1: KESTER IN VIEW OF KENNEDY DID NOT RENDER CLAIMS 1-13, 15-27, AND 29-30 OBVIOUS.	17
A. Neither Kester nor Kennedy nor their combination teaches identifying a relationship between two objects by comparing those objects, as required by the independent claims.....	18
1. Kester does not teach the claimed “comparing.”	19
2. Kennedy does not teach the claimed “comparing.”	21
3. The proposed combination does not teach or suggest the claimed “comparing.”	25
B. Petitioner fails to show that the proposed combination would meet the claimed “receiving” and “correlating” steps, as required by the	

independent claims.	27
1. Petitioner’s combination does not teach or suggest “receiving” the claimed “information about events” from remote computers, as required by independent claims 1 and 15.	28
2. The proposed combination does not teach or suggest “correlating” event data from Kennedy’s MSDM.	30
C. Petitioner fails to show that the proposed combination would have rendered the dependent claims obvious.	32
1. Petitioner’s analysis of dependent claims 2-7 and 16-21 is incompatible with its analysis of the independent claims.	33
2. Petitioner has not shown that the data used for the “comparing” includes any of the information recited in dependent claims 8 and 22.	36
VI. GROUND 2: KESTER IN VIEW OF KENNEDY AND HONIG DID NOT RENDER CLAIMS 14 AND 28 OBVIOUS.	37
VII. CONCLUSION.....	47

TABLE OF AUTHORITIES

	Page(s)
<i>Dynamic Drinkware, LLC v. Nat'l Graphics, Inc.</i> , 800 F.3d 1375 (Fed. Cir. 2015)	17
<i>Harmonic Inc. v. Avid Tech., Inc.</i> , 815 F.3d 1356, 1363 (Fed. Cir. 2016)	17

EXHIBIT LIST

2001	Declaration of Professor Nenad Medvidovic, Ph.D
2002	<i>Curriculum Vitae</i> of Professor Nenad Medvidovic, Ph.D.
2003	<i>Webroot, Inc., Open Text Inc., v. AO Kaspersky Lab et al.</i> , Case No. 6:22-cv-00243-ADA-DTG, Order Granting Third Amended Scheduling Order, Dkt. 142 (W.D. Tex., Dec. 27, 2022)
2004	<i>Sonrai Memory Ltd. v. Kingston Tech. Co. and Kingston Tech. Corp.</i> , Case No. 6:21-cv-1284-ADA, Order Denying Kingston Tech. Co. and Kingston Tech. Corp.'s Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 94 (W.D. Tex., Oct. 18, 2022)
2005	<i>mCom IP, LLC v. Cisco Systems, Inc.</i> , Case No. 6:22-cv-00261-ADA, Order Denying Defendant's Motion to Stay Pending <i>Inter Partes</i> Review, Dkt. 42 (W.D. Tex., Oct. 20, 2022)
2006	<i>Intellectual Ventures I LLC et al. v. Hewlett Packard Enterprise Co.</i> , No. 6:21-cv-00596-ADA, Order Denying Hewlett Packard Enterprise Company's Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 104 (W.D. Tex., Dec. 22, 2022)
2007	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Dec. 9, 2022 Transcript of Motions Hearing, Dkt. 133 (W.D. Tex. Dec. 15, 2022) (excerpted)
2008	Dani Kass, "Catching Up On Patent Litigation With Judge Albright" (Law360 Mar. 14, 2023); https://www.law360.com/pulse/articles/1582438/print?section=pulse/courts
2009	U.S. District Courts—Federal Court Management Statistics—Profiles—During the 12-Month Periods Ending March 31, 2018 Through 2023, "Table N/A—U.S. District Courts—Combined Civil

	and Criminal Federal Court Management Statistics (March 31, 2023)”, available https://www.uscourts.gov/sites/default/files/data_tables/fcms_na_distprofile0331.2023.pdf
2010	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Claim Construction Order, Dkt. 236 (W.D. Tex. Mar. 16, 2023)
2011	<i>Webroot, Inc. and Open Text Inc., v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.</i> , Case No. 6:22-cv-00241-ADA-DTG, First Amended Complaint for Patent Infringement, Dkt. 60 (W.D. Tex., Sept. 8, 2022) (excerpted)
2012	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Scheduling Order, Dkt. 39 (W.D. Tex. Aug. 17, 2022)
2013	<i>Webroot, Inc. and Open Text Inc., v. ForcePoint LLC</i> , Case No. 6:22-cv-00342-ADA-DTG, Defendant ForcePoint LLC’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2014	<i>Webroot, Inc. and Open Text Inc., v. Sophos Ltd.</i> , Case No. 6:22-cv-00240-ADA-DTG, Defendant Sophos Ltd.’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2015	<i>Webroot, Inc. and Open Text Inc., v. Trend Micro, Inc.</i> , Case No. 6:22-cv-00239-ADA-DTG, Defendant’s Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2016	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab.</i> , Case No. 6:22-cv-00243-ADA-DTG, Defendant’s Preliminary Invalidity Contentions Appendix A (W.D. Tex., Sept. 14, 2022) (excerpted)
2017	<i>Webroot, Inc. and Open Text Inc., v. CrowdStrike, Inc. and CrowdStrike Holdings, Inc.</i> , Case No. 6:22-cv-00241-ADA-DTG,

	Defendant's Preliminary Invalidity Contentions (W.D. Tex., Sept. 13, 2022) (excerpted)
2018	<i>Ex Parte</i> James M. Brennan, et al., Appeal 2021-003606, Application 15/285,875, Decision On Appeal Statement of the Case (PTAB Jul. 26, 2022)
2019	<i>File History</i> of U.S. Patent No. 8,418,250 ("250 File History")
2020	Declaration of Professor Nenad Medividovic, Ph.D. In Support of Patent Owner's Response, dated March 29, 2024
2021	<i>AO Kaspersky Lab v. Open Text Inc.</i> , Case No. IPR2023-00556, Deposition Transcript of Wenke Lee, Ph.D., dated March 15, 2024
2022	<i>Webroot, Inc. and Open Text Inc. v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA-DTG, Opening Claim Construction Brief From: [AO Kaspersky Lab], Dkt. 86 (E.D. Tex. Oct. 28, 2022) (excerpted)

Open Text Inc. (“Patent Owner” or “Open Text”) submits this Response to the petition of AO Kaspersky Lab (“Petitioner”) seeking inter partes review of U.S. Patent No. 8,726,389 (“the ’389 patent”).

I. INTRODUCTION

Petitioner fails to prove that any of the challenged claims would have been obvious in view of Kester and Kennedy (as proposed in Ground 1) or that dependent claims 14 and 28 were obvious in further view of Honig (as proposed in Ground 2).

First, Petitioner fails to show that either Kester and Kennedy suggest a malware classification technique that involves identifying relationships between objects by comparing information about those objects (*e.g.*, by detecting common “genes” in parent and child objects). Kester does not identify relationships between objects at all, and the “file sets” Kennedy identifies are not based on comparing information about those files. Because Petitioner makes no argument that the level of ordinary skill can fill this gap, the Board may reject the IPR challenge on this ground alone.

Second, Petitioner fails to explain how or why a skilled artisan would have combined Kester and Kennedy into a system that satisfies every limitation. Ultimately, Petitioner’s argument is only that, because Kester and Kennedy involve the general concepts of collecting and correlating data, it would have been obvious to collect the same data recited, correlate it as recited, and moreover identify

relationships using the claimed comparison. That is incorrect, and it provides a second independent reason to reject the IPR challenge.

Petitioner's arguments respecting the dependent claims (including its Ground 2 challenge) only further highlight Petitioner's failure to identify the recited "comparing" step in either reference and to explain the proposed combination. Petitioner fails to articulate a combination of the prior art that renders obvious key limitations of the '389 patented technology. The challenged claims should be confirmed as patentable.

II. BACKGROUND

A. The State of the Art.

In the field of computer security, "malware" is an executable or other computer object that contains malicious code. (EX-1001 1:19-22; EX-2001 ¶¶ 28-29.) Examples include viruses, trojans, worms, etc. (*Id.*)

A traditional approach to detecting and blocking malware was to scan computer files for "signatures" associated with *known* malware. (EX-1001 1:23-2:23.) Using this approach for *new* malware required service providers to identify the new malware, analyze it to find new signatures, and then update anti-malware programs to search for those new signatures in computer objects. (*Id.*) Another traditional approach leveraged the insight that files that had been around long enough without

behaving maliciously could be deemed safe. (EX-1001 1:48-52.)

A shortcoming of these traditional approaches was their inability to detect *new* malware quickly. The systems were *reactive*, in that they waited for malware to present itself (often by infecting networks and computers) before taking any action, or instead, waited for programs to age sufficiently without incident. (EX-1001 1:35-42, 1:53-59; EX-2020 ¶¶31-33, 44.)

To quicken the response time, some “community-based” approaches were developed, where a “central computer” gathered and used information from multiple “local” computers connected to it over a computer network. (EX-1001 1:65-2:9.) But those approaches had several shortcomings as well, including sending large files to the central computer and relying on manual analyses of the files. (EX-1001 2:9-17, 2:63-3:3.) Such manual analyses were slow, sometimes “tak[ing] days.” (*Id.*)

Accordingly, there was a long-felt need for an improved way to quickly determine whether files were safe or malware. (EX-2020 ¶¶ 29-33.)

B. The '389 Patent

The '389 patent introduced improvements in malware detection that overcame these problems in the prior art. (EX-1001 3:14-24, Claims 1, 15, EX-2020 ¶¶ 34-35.) Rather than “rel[y] solely on signature matching,” the claimed system relies on “sophisticated pattern analysis” derived from “a comparison ... between the objects

and/or their effects on the different remote computers.” (EX-1001 3:14-24).

The claimed system utilizes a community-based approach in which “remote computers” send to a centralized “base computer” certain information about computer objects stored on those “remote computers.” (*Id.* 3:3-13.) “In a preferred embodiment, the data about the computer object ... includes” (1) object data (*i.e.*, “executable instructions contained within or constituted by the object”), (2) object metadata (*e.g.*, “size,” “name,” “location”), and/or event metadata (*i.e.*, “events initiated by or involving the object when the object is created, configured or runs on the respective remote computers”). (*Id.* 3:25-36; EX-2020 ¶¶ 35-36.)

In a preferred embodiment, “the data [received about an object] is sent in the form of a key” with at least one severable “component that represents executable instructions contained within or constituted by the object.” (*Id.* 3:57-59.) “This important preferred feature allows a comparison to be made at the base computer of only the executable instructions of the object,” such that objects “that basically have the same executable instructions” are “regarded as the ‘same’ object,” even if they have different object metadata or event metadata. (*Id.* 3:59-4:12.) Thus, for example, the base computer can correlate data about two objects with the same executable instructions, even if those objects are “differently named.” (*Id.*; EX-2020 ¶ 37.)

The '389 patent discloses that embodiments may “allow[] a comparison to be

made between the objects,” such that “the comparing identifies relationships between ... objects.” (*Id.* 3:15-16, 3:37-47.) “[A]n object is considered to have a relationship based on the event performed by it upon another object, or upon it by another object,” such as “which object created this object, which objects this object created, which objects created a registry key to run or configure this object, which objects were configured by or had registry keys created by this object, etc.” (*Id.* 11:7-13.) Moreover, relationships may depend on whether two objects share certain “genes”—*e.g.*, “[t]he key relating to [a] first object may be inspected to identify a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object.” (*Id.* 17:22-18:15; EX-2020 ¶¶ 37-38.)

“[T]o establish if an object is safe, unsafe or suspicious,” the data about the object is “captured and correlated from all of the remote computers that have seen the object,” relationships to other objects are identified, and “heuristics” are applied. (*Id.* 10:41-11:4.) These heuristics can “utilize[] an object’s relationship to other objects that act upon it or upon which it acts.” (*Id.* 11:5-60.) For example, a child process that inherited certain genes may also inherit its parent’s classification, and if the parent’s or child’s subsequent behavior reveals it is malware, the other (and all siblings) can be marked as malware too. (*Id.* 17:43-51.) Similarly, the system can

monitor “twins of objects (*e.g.*, the same process running on other systems in the community) to compare the pattern of their issue and to make decisions as to whether or not to treat any particular process as malware.” (*Id.* 18:3-7; EX-2020 ¶¶ 38-39.)

By collecting, correlating, and comparing data about objects from many computers, the claimed system can create “a complex series of correlation points” that “allow[s] ancestral relationships to be considered.” (*Id.* 11:13-17.) And because “a typical implementation in the field” may include “several thousands or even millions of remote computers connected or connectable to the base computer,” “any objects that are newly released upon the Internet or that otherwise are found on any of these remote computers will soon be found and signatures created and sent to the base computer by the respective remote computers.” (*Id.* 8:56-63; EX-2020 ¶ 40.)

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

Claim 1 recites:

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

comprising:

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

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

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

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

[1.5] comparing, by the base computer, the correlated data about the computer object received from the first and second remote computers to other objects or entities to identify relationships between the correlated data and the other objects or entities; and

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

(*Id.* 20:27-21:3; EX-2020 ¶¶ 40-42.)

An example embodiment of the claimed method may work as follows. A base computer receives data from a first and second computer about an Object X. The first computer reports that X was created by Object PARENT, and the second computer reports that X created Object CHILD. The base computer correlates the data from the first and second computer as being about Object X. The base computer then *compares* CHILD to another object in the community (Object MAL) to identify their

relationship. For example, the base computer determines that CHILD is MAL's parent, in part by comparing the keys for CHILD and MAL to determine that the two share a "gene." Based on that comparison, the base computer classifies Object X (and CHILD and PARENT) as malware. (EX-2020 ¶¶ 43-44.)

C. Claim Construction

Petitioner relies on the plain and ordinary meaning of the claims. (Petition 6; EX-2020 ¶ 45.) In the parallel district court litigation, Petitioner and its co-defendants argued for constructions of various terms, but the district court construed each to have its plain and ordinary meaning, and it made the following clarifying remarks:

Terms	Construction
"remote computer" "first remote computer" "second remote computer" "similar objects" "similar computer objects"	Plain-and-ordinary meaning (EX-2010 3, 5, 8.)
"events initiated [by] or involving the computer object"	Plain-and-ordinary meaning Note: the plain-and-ordinary meaning of "events" includes but is not limited to "actions or behaviours of an object acting upon another object or some other entity" (<i>Id.</i> 5.)

“correlating...at least a portion of the data about the computer object...comparing...the correlated data about the computer object”	Plain-and-ordinary meaning. Not <i>all</i> of the data must be correlated before comparing. (<i>Id.</i> 6-7.)
“a mask for said computer object that defines acceptable behaviour for the computer object”	Plain-and-ordinary meaning. The claim language recites that at least a portion of the data, but not all of the data, must be correlated before comparing. (<i>Id.</i> 6-7.)

D. The Level of Ordinary Skill

A POSITA at the time of the invention of the '389 patent (which claims priority to an earlier application (GB 0513375.6) filed in Great Britain on June 30, 2005) would have had a Bachelor's degree in an accredited program of Electrical Engineering, Computer Engineering, or Computer Science or in a similar discipline, and have 2-3 years of practical work or research experience in the general fields of electrical engineering, computer science, networking, communications, and device and network security. (EX-2020 ¶¶46-50.) More advanced degrees and/or training in a related discipline can compensate for shorter work experience. (*Id.*)

III. OVERVIEW GROUNDS AND CITED ART.

Petitioner has raised the following two grounds of obviousness:

Ground 1	Claims 1-13, 15-27, and 29-30	§ 103: Obvious over Kester in view of Kennedy
Ground 2	Claims 14 and 28	§ 103: Obvious over Kester in view of Kennedy in further view of Honig

A. Kester

Kester discloses “[a] system and method for updating, monitoring, and controlling applications on a workstation.” (EX-1004 Abstract; EX-2020 ¶¶ 51-74.)

In Kester, each “workstation includes a workstation management module” that “detect[s] the launch of an application on the workstation and determines [] access privileges for the workstation and/or user.” (*Id.* Abstract, [0040].) An application is identified using a “hash” of its binary. (*Id.* [0052]). The application’s “access privileges” can be associated directly with that binary (*e.g.*, in a “hash/policy table”) or with a “category” assigned to the application. (*Id.* [0040].) One “category” disclosed in Kester is “malware.” (*Id.* Figure 4A; EX-2020 ¶¶ 54-56.)

If “the application is allowed to run on the workstation,” the “workstation management module” then “monitor[s] the ongoing network activity or behavior of

the application.” (*Id.* [0041].) “For example, each time an application ... attempts to access a network, the workstation management module determines whether to allow [the action].” (*Id.*) Again, access privileges can be associated with the application directly or they can be applied based on the application’s category. (*Id.* [0048]; EX-2020 ¶ 57.)

The application’s category is assigned manually by a human “network administrator” using a “classification user interface.” (*Id.* [0064]. [0067], [0069], [0071]-[0073], [0076], [0134-136].) The interface may be provided by a server-side component, such as an “application sever module” or “application database factory,” which then communicates each application’s category to the workstations for implementation. (*Id.* [0064]. [0067], [0069], [0077]; EX-2020 ¶¶ 63, 73.)

Using the monitored “network activity,” the workstations and server-side components can learn each application’s “*expected* network behavior.” (*Id.* [0043]-[0045].) The “expected network behavior” can be based on the prior access patterns of the application (or of a related application, such as an earlier version) on the workstation or across multiple workstations. (*Id.* [0043]-[0045], [0069]-[0070].) The workstations capture and log those prior access patterns in a “logging database” and report that data to the server-side components. (*Id.* [0045], [0051], [0062], [0069]-[0071]; EX-2020 ¶ 63)

Kester teaches that determining the expected network activity may involve “merging and sorting” the data about the application received from the different workstations. (*Id.* [0074]. [0087].) Kester provides only two examples of “merging and sorting”: (1) merging and sorting network access data based on “destination IP address” (*id.* [0162]); and (2) merging and sorting application access data based on an application’s “request frequency” (*id.* [0119]; EX-2020 ¶¶ 64-72).

If a server-side component (*e.g.*, “application server module”) determines that “[a] categorized application ... does not operate in a predetermine manner,” it may place the application in an “uncategorized application database.” (*Id.* [0070].) The human network administrator then reviews data about the uncategorized application to categorize it. (*Id.* [0071]-[0072], [0134]-[0136]; EX-2020 ¶ 73.)

B. Kennedy

Kennedy discloses a method of “detecting malicious software through file group behavior.” (EX-1005 Title.) Rather than a community-based approach (where threat data is aggregated from many computers on a network), Kennedy utilizes a local approach where a single computer uses a “malicious software detection module (MSDM)” to identify suspicious files, monitor their behavior, and ultimately categorize them as malicious. (*Id.* 1:42-43; EX-2020 ¶¶ 75-81.)

Kennedy discloses that the MSDM includes (1) an “arrival module” for

“detect[ing] the arrival of new software” on the computer, (2) a “file set tracking module” for identifying sets of files associated with that new software, and (3) a “file set behavior monitoring module” for “determin[ing] whether the suspicious software represented by each set is malicious.” (*Id.* 1:42-43, 3:47-59, 4:19-63, 4:64-5:42.)

The “arrival module” “monitors the storage device [of the computer] and detects the arrival of new software ... via one or more entry points, or ‘portals.’” (*Id.* 3:3:47-59.) If software arrives by a “suspicious” portal (*e.g.*, an “open drive share portal”), the MSDM marks each file of that software as “suspicious.” (*Id.* 4:5-18; EX-2020 ¶ 76.)

The “file set tracking module” tracks the “suspicious software” to identify “a set of files” associated with that software. (*Id.* 4:19-20.) Initially, “[t]he set [of files] includes each file that constitutes the suspicious software.” (*Id.* 4:22-23.) The “file set tracking module” then “tracks the behaviors of the files in the [suspicious] set and records each time a file in the set ‘touches’ another file on the storage device or elsewhere in the computer system.” (*Id.* 4:23-26.) To “touch” can “mean[] to create, modify, and/or read a file,” or to “fewer and/or more actions,” such as “creating and modifying an executable file and/or a file over a specified size.” (*Id.* 4:26-34.) Each time a file is touched, “[t]he file set tracking module adds the touched file to the set of the file that touched it.” (*Id.* 4:33-34.) Thus, the system “creates a logical set of

files associated with the suspicious software,” where “[t]he set contains files that form the suspicious software and any file that is touched by a file already in the set.” (*Id.* 5:57-63; EX-2020 ¶ 78.)

Finally, the “file set behavior monitoring module” “monitors the behaviors of the files within the set associated with each suspicious software,” and it uses certain behavioral “heuristics to determine whether the suspicious software represented by each set is malicious.” (*Id.* 4:64-5:42.) For example, according to one heuristic, the “software is malicious if it attempts to copy itself to another computer system via a technique such as email and/or an open drive share.” (*Id.* 5:13-17.) In another example, “software that sends more than a certain number of emails (e.g., N=10) within a certain time period is malicious.” (*Id.* 5:23-25). “Other heuristics specify that software is malicious if it mines files on its local computer system for email addresses, performs certain network activities such as enumerating for open file shares and/or copying files, makes registry changes related to the persistency of the Software, *etc.*” (*Id.* 5:29-33; EX-2020 ¶¶ 79-81.)

C. Honig

Honig discloses a computer system architecture that supports the automation of certain steps in deploying “data-mining” intrusion detection systems (IDSs). (EX-1005 4:53-56; EX-2020 ¶ 82) Data-mining IDSs collect data from sensors

monitoring a system and use machine learning algorithms on a large set of “training data” to build detection models. (EX-1005 2:5-16; EX-2020 ¶¶ 82-84.)

The type of training data needed depends on the type of detection model (and, therefore, the type of machine learning algorithm) that used to detect intrusions. (EX-2020 ¶ 83.) Honig discloses at least three general types of model generation algorithms: (1) “misuse detection,” which models known attack behavior and trains on labeled normal and attack data; (2) “supervised (traditional) anomaly detection,” which detects activity that diverges from a model of normal activity and trains on normal data; and (3) “unsupervised anomaly detection,” which also detects activity that diverges from a model of normal activity but which trains on unlabeled data. (EX-1005 2:29-49, 20:33-38; EX-2020 ¶¶ 83-84.)

Honig's architecture consists of: sensors and detectors for collecting information from the monitored system; a data warehouse which stores the collected data; model generators which access the data and train data mining-based detection models; model distributors which transfer the models to detectors; and analysis engines to provide data analysis capabilities. (EX-1005 6:64-7:7, 1:44-47, 4:53-56, 3:24-30; EX-2020 ¶ 84.)

IV. LEGAL STANDARD

“In an inter partes review, the burden of persuasion is on the petitioner Failure to prove the matter as required by the applicable standard means that the party with the burden of persuasion loses on that point—thus, if the fact trier of the issue is left uncertain, the party with the burden loses.” *Dynamic Drinkware, LLC v. Nat’l Graphics, Inc.*, 800 F.3d 1375, 1378-79 (Fed. Cir. 2015); *Harmonic Inc. v. Avid Tech., Inc.*, 815 F.3d 1356, 1363 (Fed. Cir. 2016) (noting petitioners have the “burden from the onset to show with particularity why the patent it challenges is unpatentable”).

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

In Ground 1, Petitioner proposes a combination of Kester and Kennedy whereby (1) Kester’s workstations use Kennedy’s “file set tracking module” to identify “file sets” of related files, (2) those workstations then send data about those file sets and their behaviors to Kester’s “application server module,” and (3) Kester’s “application server module” uses Kennedy’s “behavior monitoring module” to apply Kennedy’s classification heuristics to the received data, thus automatically classifying certain files as malware. (Petition 28-29; EX-2020 ¶ 88.)

Petitioner’s combination suffers from two shortcomings. *First*, neither of the

references (alone or in combination) teaches or suggests the claimed idea of identifying relationships between objects based, at least in part, *on a comparison of the objects* (e.g., detecting common “genes” in parent and child objects). *Second*, Petitioner discusses each claim limitation with respect to only one reference in isolation, failing to explain how (or why) *the combination* satisfies every limitation, including the “receiving,” “correlating,” and “comparing” steps. The Board may deny the petition for either reason (or both). (EX-2020 ¶ 89.)

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

One shortcoming of Petitioner's combination is that neither Kester nor Kennedy—alone or in combination—teaches or suggests the idea of “identifying relationships” between two objects by “comparing” information about those objects.

The '389 patent discloses that the invention “tracks which objects are related to each other and uses the concept of ancestry to enable objects to be marked as malware.” (EX1001 17:22-24.) It further discloses that whether two objects are “related” may depend on event metadata (e.g., that a “process may spawn child processes which are therefore related”) and on object data, such as shared “genes” (i.e., “a component that uniquely identifies the first object and that is inherited or otherwise present in the key of a descendant or other related object of the first object,”

“referred to herein as a ‘gene’”). (*Id.* 17:26-30; EX-2020 ¶¶ 90-91.)

The challenged claims recite this idea of determining relationships between objects by *comparing* data about those objects. Specifically, they require “comparing ... the correlated data about the [first] computer object ... to other objects or entities to identify relationships between the correlated data and the other objects or entities.” (*Id.* 20:56-60.) This recited comparison “to other objects or entities” invokes procedures described in the specification, such as comparing two objects’ respective object data to determine whether they share a given “gene” and are therefore related. (*Id.* 17:22-31; EX-2020 ¶ 92.)

Neither Kester nor Kennedy teaches or suggests this limitation. Petitioner’s position is unclear because, at different points, Petitioner seems to argue that the “comparing” limitation is taught either by Kester’s comparison of an application’s behavior to that *same application’s* expected behavior (Petition 26) or instead, by Kennedy’s determination that an object is related to known malware (*id.* 28-29.) Regardless, neither of these teachings, alone or in combination, discloses the claimed comparing step. (EX-2020 ¶¶ 94-100.)

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

Petitioner’s first argument is that Kester teaches the “comparing” limitation by comparing an application’s behavior to the *same application’s* expected behavior.

(Petition 26.) This argument is wrong for at least two reasons. (EX-2020 ¶ 94.)

First, rather than a comparison of data about an object “to other objects,” this purported “comparing” in Kester is a comparison of data about an object to other data about *the same* object. That is, Petitioner’s argument is that Kester compares an application’s behavior to the *same application’s* expected behavior. (*Id.*) In contrast, the claims recite comparing data about a computer object “to *other* objects or entities”—not only to the *same* application. (EX 1001 20:56-60; EX-2020 ¶ 95.)

Second, Kester’s comparison of an application’s behavior to that same application’s expected behavior is not made “to identify relationships” between those two objects, as claimed. Indeed, Petitioner does not even present that argument. Nor could it because this “comparing” Petitioner relies on is of data about two objects whose relationship is already known: *they are the same application*. Thus, the “comparing” on which Petitioner relies in Kester is not done “to identify relationships” between objects. (EX-2020 ¶ 96.) Rather, it is done to determine whether the application is behaving in an expected way.

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

identified above. (EX-2020 ¶ 97.)

As to the first shortcoming (*i.e.*, that Kester compares data only about the *same application*), even if Kester determines that application's expected behavior in part based on a related application's behavior, the resulting expected behavior is still data about *the application*, not about the related application. Thus, comparing that data (about the application) to other data about that same application is not comparing data about the application "*to other* objects or entities," as claimed. (EX-2020 ¶ 98.)

As to the second shortcoming (*i.e.*, that Kester's comparison of an application's behavior to that same application's expected behavior is not made "to identify relationships" to other objects or entities), Petitioner's argument is inapplicable. Kester's comparison involves data about only one application, and it is made to determine whether that application is behaving in an expected manner, not to determine whether the application is *related to* any other application. (EX-2020 ¶ 99.)

Thus, Petitioner makes no plausible argument that Kester teaches the claimed idea of "comparing" information about two different objects to identify relationships between those objects. (EX-2020 ¶ 100.)

2. Kennedy does not teach the claimed "comparing."

Unlike Kester, Kennedy does teach the idea of a malware classifier that identifies relationships between different objects (which it uses to create the disclosed

“file sets”). (EX1005 4:19-63.) But Kennedy identifies those relationships based only on events—*i.e.*, detecting that one object “touched” another—not based on a comparison of information about two different objects, as claimed. (*Id.*) Thus, Kennedy does not disclose the claimed “identify[ing] relationships” to “other objects or entities” by “comparing” data about a first object “to other objects or entities.” (*Id.* 20:56-60; EX-2020 ¶¶ 101-112.)

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

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

(Petition 29; EX-2020 ¶ 102.)

At his deposition, Petitioner’s expert (Dr. Wenke Lee) confirmed that this is indeed Petitioner’s theory. Asked repeatedly how the combination satisfies the claimed “comparing” limitation, Dr. Lee testified each time that the combination performs the comparing step by comparing unspecified data about the object being evaluated against unspecified data about other objects that the system has already

determined are malware, and, if the data are sufficiently similar, classifying the object as malware. (EX-2021 160:16-162:12; 174:2-175:19; 188:9-190:20.)

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

First, Kennedy does not teach or suggest, as Petitioner argues, the idea of comparing data about the object being evaluated against data about other objects that the system has already determined are malware. Rather, to determine whether a file is malware, Kennedy applies behavioral heuristics that compare an object's behavior to pre-known behavioral patterns of malware. For example, Kennedy discloses the following behavioral heuristics for determining whether software is malware: the software (1) "attempts to copy itself to another computer system via a technique such as email and/or an open drive share," (2) "sends more than a certain number of emails (e.g., N=10) within a certain time period," or (3) "mines files on its local computer system for email addresses, performs certain network activities such as enumerating for open file shares and/or copying files, makes registry changes related to the persistency of the Software, *etc.*" (EX 1005 5:4-33) None of those heuristics involve "comparing" the software's behavior to other software that the system

previously determined was malware. Thus, Kennedy does not teach or suggest “comparing ... data about the computer object ... to other objects or entities,” as claimed.

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

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

meaning of the claims.

Thus, Petitioner fails to show that Kennedy teaches the claimed “comparing” of objects and entities to identify relationships between them.

3. The proposed combination does not teach or suggest the claimed “comparing.”

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

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

Moreover, Petitioner's argument regarding the *proposed combination* of Kester and Kennedy reveals that any purported “comparing” in Kester is irrelevant. Kester's purported “comparing” (*i.e.*, of an application's behavior to its expected behavior) is part of Kester's manual classification procedure. Specifically, Kester compares the application's behavior to the application's expected behavior to determine

whether the application is not operating “in a predetermined manner” and to consequently place the application in an “uncategorized application database” for later manual classification. (EX1004 [0070].) But Petitioner’s *proposed combination* does not use Kester’s classification technique. Instead, the proposed combination classifies an application automatically, *using Kennedy’s behavioral heuristics* rather than manually, using Kester’s classification approach. (Petition 28-29, 32) (“In this combination, Kester’s application server ... utilizes Kennedy’s automated heuristics.”) If the proposed combination does not use Kester’s classification technique, then it does not use Kester’s purported “comparing” (which is part of that technique). Thus, the purported “comparing” Petitioner identifies in Kester is irrelevant to whether the *proposed combination* of Kester and Kennedy renders the claims obvious. (Petition 26.)

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

thus does not use Kester's classification technique, including its purported "comparing." The "comparing" Petitioner purports to identify in Kester is therefore irrelevant to whether that proposed combination rendered the claims obvious.

* * *

In sum, Petitioner has failed to show that either Kester, Kennedy, or the combination thereof reaches or suggests the claimed "comparing step.

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

A second independent reason to find the challenged claims patentable is that Petitioner does not sufficiently explain the proposed combination of Kester and Kennedy, including how that combination supposedly satisfies every limitation of the independent claims. Petitioner tries to identify each limitation in one reference or the other, but when it comes to combining those references, Petitioner leaves the Board (and Patent Owner) guessing about how the purported combination might work and why.

Consequently, it is unclear how the proposed combination would work, why a POSITA would have been motivated to make it, or how it would satisfy all claimed limitations, including the claimed (1) receiving, (2) correlating, and (3) comparing steps. We have already discussed above (in Section V.A.3) a shortcoming of the

proposed combination respecting the claimed “comparing” step. Below, we highlight further shortcomings and ambiguities regarding the claimed “receiving” and “correlating” steps.

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

Petitioner argues that it would have been obvious for Kester’s “application server module” to receive data from Kennedy’s MSDMs, executing on multiple workstations. (Petition 29.) But Petitioner does not say *what data* each MSDM supposedly captures and sends to the “application server module.” This omission creates a threshold problem: Petitioner fails to show that it would have been obvious for Kennedy’s MSDM to capture and send the claimed “information about events,” including “an object initiating the event, the event type, and an identity of an object or other entity on which the event is being performed,” as required by independent claims 1 and 15.

Petitioner argues that *Kester* (alone) teaches that the “application server module” receives “information about events”—specifically, information indicating that a given application (actor object) has attempted to access (event type) a particular network address (target “entity”). (Petition 13-19.) *But this is not the data Petitioner argues would be “correlated” and “compared” to “classify” an object as malware.*

(Petition 27.) Instead, Petitioner asserts that those steps would be performed on data sent *from Kennedy's MSDM*, as it operates on Kester's workstations. (Petition 27-32.) Thus, for Petitioner combination to render the claims obvious, Petitioner must show that *Kennedy's MSDM* captures the claimed information about events or that it would have been obvious to modify the MSDM to do so and to send that information to Kester's server-side components. But Petitioner does neither.

For instance, Petitioner asserts that each workstation would include Kennedy's "file set tracking module" (Petition 28), but that module merely identifies "file sets," not the "information about events" recited in the claims. Thus, adding Kennedy's "file set tracking module" to Kester's workstations would not enable those workstations to capture (let alone send) the claimed "information about events."

Even if Kennedy's "file set tracking module" does record (at least temporarily) "each time a file in the set 'touches' another file" (EX1005 4:19-36.), Kennedy does not teach that the "touch" data captured by the file set tracking module includes the claimed "information about events" (*e.g.*, includes "event type"). Nor does Kennedy teach that this "touch" data is ultimately used by Kennedy's classification heuristics. To the contrary, Kennedy teaches that those heuristics classify files based on data captured by a *different* module—*i.e.*, the "file set behavior monitoring

module”—which Petitioner has proposed would be placed on Kester’s *servers*, not on Kester’s workstations. (*Id.* 4:64-5:3.) In short, the information captured by the “file set tracking module” does not disclose the claimed “information about events.”

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

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

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

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

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

Even under those assumptions, however, the petition leaves unexplained *why* and *how* a skilled artisan would apply Kester's "merging and sorting" techniques to Kennedy's data. Consider Kennedy's heuristic that software is malicious if it "sends more than a certain number of emails (e.g., N=10) within a certain time period." (EX1005 5:4-34.) The data for that heuristic is presumably the number of emails each application sends over time. How (and why) would a POSITA apply (to that data) Kester teaching of "merging and sorting" network access events by "destination IP address?" (EX 1004 [0162].) Petitioner does not say.

Instead, Petitioner's position appears to simply be that, because Kester applies certain merging and sorting techniques to certain data for a certain purpose, it would have been obvious for the proposed combination to apply *any* merging and sorting

techniques to *any* other data for *any* other purpose. This is insufficient to set out a proposed combination of references, let alone to explain why a POSITA would have been motivated to make that combination.

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

* * *

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

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

Petitioner's analysis of the dependent claims exposes further inconsistencies in how Petitioner has described the proposed combination. For example, while

Petitioner's analysis of the independent claims relies on relationships purportedly identified by "comparing" a file to known malware, its analysis of many dependent claims instead relies on *other* relationships, such as those identified based on events rather than by any purported "comparing" or data about different objects. These types of oversights provide further reason to reject Petitioner's arguments respecting the dependent claims.

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

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

Dependent claims 2-7 (and corresponding claims 16-21) all concern the identified "relationships" between objects and how those relationships are used in malware classification. The claimed details about the relationships include that they are either "direct" or "via other objects" (claims 2 and 16), that they are parent, child,

other process relations (claims 7 and 21), and/or that a related object is the object or similar object stored on the first or second computers (claims 6 and 20). The claimed details about how those “relationships” are used include the ideas of monitoring related objects “as a homogeneous entity” (claims 3 and 17), deducing that an object is malware based on the behavior of the related object (claims 4 and 18), and classifying the object as malware if the related object is malware (claims 5 and 19).

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

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

And yet, when the time comes to show that those same “similarity” relationships (purportedly derived from the claimed “comparing ... to identify

relationships”) have the characteristics recited in the dependent claims (*e.g.*, are parent/child) or that they are used in the ways recited in those claims (*e.g.*, to trigger monitoring of the related objects as a “homogenous entity”), Petitioner pivots to arguing that the “relationships” are something different altogether: Kennedy’s “file set” relationships.

The similarity “relationships” used in the Petitioner’s analysis of the independent claims cannot satisfy the limitations of dependent claims 2-7 (and 16-21) because they are simply incompatible. The similarity “relationships” are that an object behaves similarly to known malware. To the extent that this is a “relationship” within the meaning of the claims (it is not), the relationship is not one that can be described as “direct” or “via other objects” (claims 2 and 16), a “parent,” “child” or “otherwise process-related” (claims 7 and 21), or to the object or similar object stored on the first or second computers (claims 6 and 20). Moreover, the claimed references do not suggest that, based on these similarity relationships, an object that was labeled as malware for being similar to known malware would be monitored with that known malware “as a homogeneous entity” (claim 3), be marked as malware based on the subsequent behavior of the malware (claim 4), or be classified as malware if the known malware is malware (claim 5). (EX-2020 ¶ 142.)

Conversely, the Kennedy “file set” relationships Petitioner uses to analyze the

dependent claims are not identified by the claimed “comparing.” Petitioner’s analysis of the independent claims includes no argument that the purported “comparing” creates Kennedy’s “file set” relationships. No such argument is possible because Kennedy’s “file set” relationships are based on sets of files that suspicious software “touched,” not on comparing two files.

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

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

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

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

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

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

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

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

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

However, Kester's expected network activity—contained within its hash/policy table—is not a mask that "defines acceptable behavior" because the hash/policy table and the expected network activity identified within that table includes unacceptable behavior. (*See* EX-1004 [0070], [0185]; EX-2020 ¶¶ 151-152.)

For example, Kester teaches that when an application is not operating in a "predetermined manner," it is placed in the uncategorized applications database for

future review, such that those attributes might be added to the hash/policy table. (EX-1004 [0070]; EX-20209 ¶ 152.) That is, whether an application is operating in a “predetermined” or “expected” manner is dependent on whether that activity is contained within the hash/policy table and not whether that activity is acceptable or unacceptable. (EX-2020 ¶ 152.) Petitioner’s discussion of the combination does not address this deficiency in any way, let alone provide any explanation for why a skilled artisan would have found this limitation obvious in light of both references. Thus, Petitioner has failed to show that the combination of Kester and Honig would have rendered the independent claims obvious. (EX-1001 cls. 14, 28.)

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

application. Using evidence from hints and other research, determine allowed behavior for each application.”), [0089]-[0090], [0187].)

Claims 14 and 28 rely on a mask that defines the boundaries of an object's behavior by defining acceptable behavior. (EX-1001 cl. 14 (“generating a mask for said computer object that defines acceptable behaviour for the computer object, monitoring an operation of the computer object on at least one of the first and second remote computers and reclassifying the computer object as malware if the actual monitored behaviour extends beyond that permitted by the mask.”) cl. 28 (“generate a mask for a computer object that is initially classed not as malware, said mask defining acceptable behaviour for the computer object”).) The invention limits the mask to acceptable behavior, such that future object behavior can be allowed or disallowed based on whether it falls within or outside of the mask. (Section II(B), *supra*; EX-2020 ¶ 154.) The benefits of using a generated mask that defines acceptable behavior are that: (1) the invention avoids the delay associated with waiting for a human to review each object and designate whether specific behaviors should be permitted or not; and (2) when an object's behavior extends beyond the mask, the invention knows that the object should be reclassified from “not malware” to “malware.” (EX-2020 ¶¶ 154-155.)

Kester fails to disclose generating a mask and fails to disclose a mask that

defines only acceptable behavior. As discussed above, the challenged claims recite generating a mask and using a mask that defines acceptable behavior to allow or disallow an object to run. In contrast, Kester relies on a human to pre-review each application and/or behavior and designate whether each application/behavior should be permitted or not before the hash/policy table may be used to monitor it. (*Id.*; EX-2020 at 8:3-20.) When Kester's system encounters a new behavior that has not yet been reviewed, that behavior is logged to a database to eventually be reviewed by a human reviewer. (EX-1004 [0059], [0144]-[0145].)

To argue that Kester discloses a mask that defines acceptable object behavior, Petitioner asserts that Kester discloses "expected or allowed network activity (also referred to as expected or predetermined behavior)." (Petition 57.) But Petitioner wrongly equates "expected" network activity with "allowed network activity." Kester teaches that expected network activity may be allowed or disallowed based on the human reviewer's determination. (E.g., EX-1004 [0185].) Specifically, Kester teaches that because "network attributes associated with the application may be different for each attempted access," its system "may allow a first combination of one or more network attributes while disallowing a second combination of the one or more network attributes." (EX-1004 [0185], [0083].) Kester describes network activity as "expected" or "predetermined" to indicate that a human reviewer has

previously reviewed the specific combination of network attributes and that a predetermined association with a policy (to allow or not allow) has been stored in the hash/policy table. (EX-1004 [0057]-[0059] (“If the execution launch detection module...does not find the application hash in the hash/policy table [] (for example, the application is uncategorized or the application is behaving unexpectedly)...”).)

That is, an application operating in an expected or predetermined manner—*i.e.*, within the boundaries of Petitioner’s alleged “mask”—might still be disallowed from continuing to run based on the associated policy. (EX-2020 ¶¶ 157-159.) This result is intuitive based on Kester’s goal—*i.e.*, “to manage the broader intersection of employees with their computing environments” on a workstation-by-workstation basis. (EX-1004 [0006], [0115]-[0116].) Within this context, Kester’s system seeks to provide a different type of protection that is distinct from that of the ’389 patent.

For example, an employer might wish to limit particular activity to management only. (EX-2020 ¶¶ 157-161.) Following Kester’s approach, a human administrator reviews the application and associated network access data. (*Id.*) The human administrator then determines that, where the network access attributes indicate that a manager’s workstation has made the request, the request should be associated with a policy that allows the request. (*Id.*) But if a non-manager’s workstation makes the request, the human administrator determines that such attributes should be

associated with a policy that does not allow the request. (*Id.*) Kester teaches that a hash/policy table will be generated specific to the requesting workstation and download it to the workstation. (EX-1004 [0116].) Thus, while the particular activity, *e.g.*, the network access request, may be expected or predetermined such that it is included in the hash/policy table, it may be unacceptable and will not be allowed. (*Id.*, EX-2020 ¶¶ 131-132, 135-136.)

Accordingly, Kester relies on a hash/policy table containing “expected” activity reviewed by a human administrator that may include acceptable activity but that also includes unacceptable activity. Under the district court’s claim construction—which Petitioner urged in the parallel litigation—the claimed mask must define only acceptable behavior, and acceptable behavior cannot be defined by the inverse of unacceptable behavior. *See* Section II(C), *supra*. Thus, Kester does not disclose a mask that defines acceptable behavior of the object.

As a result of this critical distinction, Kester’s “expected network activity” cannot provide the same benefits as the ’389 patent’s claimed mask. (EX-2020 ¶¶ 151-160.) For example, the ’389 patent allows an object to run when its monitored behavior falls within the object’s mask:

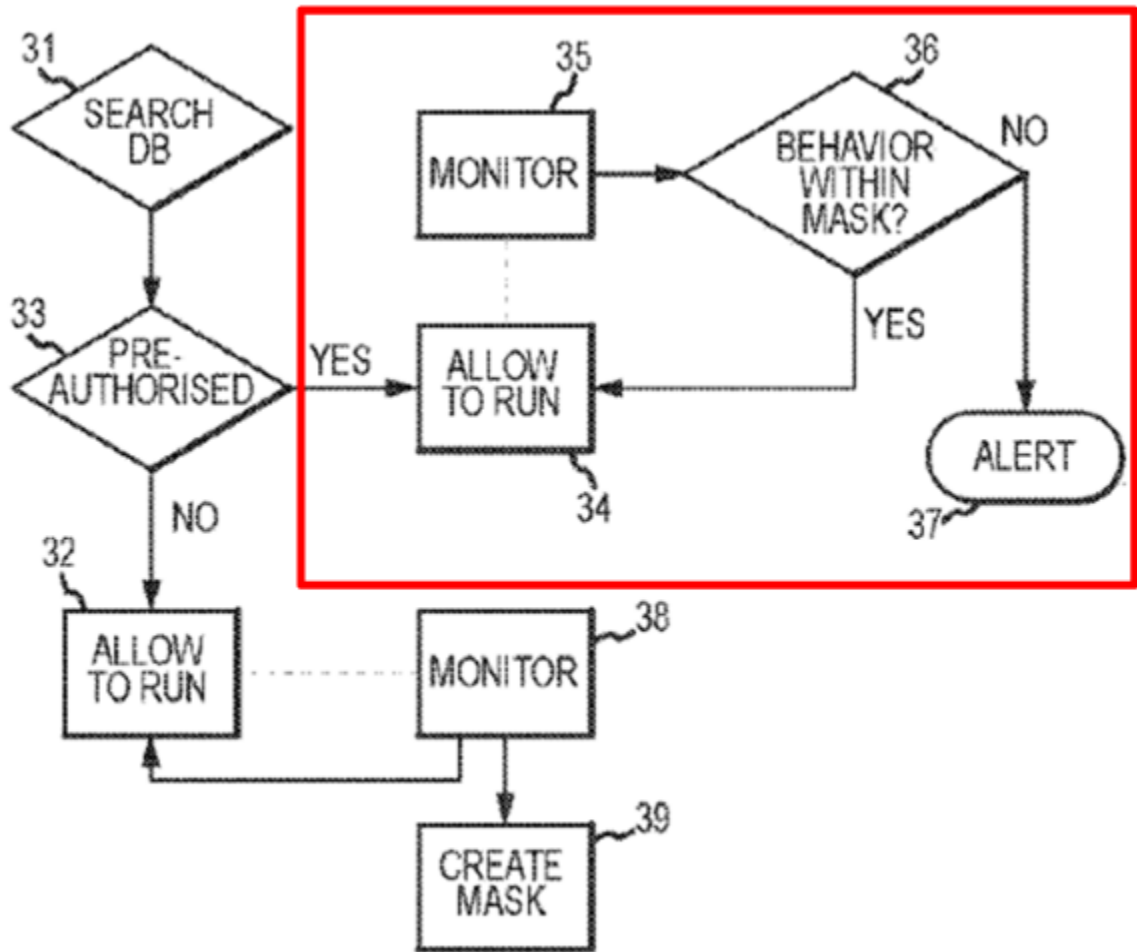


FIG.3

(EX-1001 Fig. 3 (annotated); *id.* cls. 14, 28.) And when that behavior extends beyond the mask, the object is not allowed to run and is reclassified as malware. (EX-1001, cls. 14, 28.) But because Kester's expected network activity contains both acceptable and unacceptable activity, an application whose behavior is within this alleged mask cannot categorically be allowed to run. (EX-1004 [0042]-[0043], [0185]-

[0186].) To do so would contradict Kester's purpose, which is to apply predetermined policies on a workstation-by-workstation basis across all expected activity, thereby controlling employees' interaction with the internet. (EX-1004 [0006], [0042]-[0043]; EX-2001 ¶¶ 159-160.)

As another example, because Kester's expected network activity defines unacceptable behavior, it cannot be used as a tool for informing when an application must be reclassified as malware. (EX-2020 ¶ 161.) Within the context of Kester's disclosure, when an application is operating unexpectedly—*i.e.*, its hash does not match an existing hash in the hash/policy table—that merely means that the application or the application's particular activity has not yet been reviewed, categorized, or associated with a policy by an administrator. (EX-1004 [0059].) Thus, the application—whether previously categorized or uncategorized—is added to the “uncategorized applications database” and subsequently reviewed by an administrator. (EX-1004 [0070]-[0071].) However, the fact that the activity-in-question was unexpected, *i.e.*, previously unreviewed, does not mean that it is necessarily malicious or that the application should be categorized as malicious. (EX-2020 ¶ 161.) Again, a human administrator will be required to review the application and the network attributes associated with its activity to determine whether such activity should be permitted or not, as well as whether the application should be categorized and/or

recategorized. (EX-1004 [0070]-[0071]; EX-2010 ¶ 161.)

Because Kester's hash/policy table includes unacceptable expected network activity and because a human must determine expected network activity, Kester's teachings do not practice or suggest the challenged claim limitations, and a POSITA could not achieve the same benefits of the claimed invention. (EX-2020 ¶¶ 161-167.) Accordingly, Petitioner has not shown (and cannot show) that Kester discloses or suggests generating a mask that defines acceptable behavior as recited in claims 14 and 28 claims of the '389 patent.

While Petitioner asserts that a POSITA would have been motivated to automate the human reviewer's determination of expected network activity in Kester's system based on Honig's teachings (Petition 58), this does not address the deficiency already present in Kester. Even if a POSITA were motivated to attempt to automate Kester's determination of expected network activity (which he would not be, as discussed further below), the discussion above shows that Kester's expected network activity is not a mask that defines acceptable behavior as claimed. *See* Section III.A, *supra*. Thus, automatically generating Kester's expected network activity still would not practice the method or achieve the same benefits of the challenged '389 patent claims. Petitioner does not address this deficiency or argue that the prior art would have rendered such a mask obvious. Accordingly, Kester and Honig do not render

any of the challenged claims obvious.

VII. CONCLUSION

For the foregoing reasons, the Board should hold that the Petition fails to demonstrate, by a preponderance of the evidence, that any of the challenged claims are unpatentable.

Date: March 29, 2024

Respectfully submitted,

BY: /s/ Brian Eutermoser
Brian Eutermoser
Registration No. 64,058

CERTIFICATION OF WORD COUNT

The undersigned hereby certifies that the portions of the above-captioned **PA-
TENT OWNER'S RESPONSE** specified in 37 C.F.R. § 42.24 includes 9,548 words and therefore complies with the 14,000-word limit set forth in 37 C.F.R. § 42.24. This word count was prepared using Microsoft Word 365 v.2202.

Date: March 29, 2024

Respectfully submitted,

BY: /s/ Brian Eutermoser
Brian Eutermoser
Registration No. 64,058

CERTIFICATE OF SERVICE

The undersigned hereby certifies that the foregoing **PATENT OWNER'S RESPONSE** and all supporting **EXHIBITS 2020-2022** were served by filing the documents through the Patent Trial and Appeal Case Tracking System (P-TACTS) and courtesy copies were also served via electronic mail, in their entireties on the following:

Eric H. Chadwick
Erin E. Block
Joseph T. Miotke
DeWitt, LLP
901 Marquette Avenue
2100 AT&T Tower
Minneapolis, MN 55402
ehc@dewittllp.com
eblock@dewittllp.com
jtm@dewittllp.com
docketing@dewittllp.com

*Counsel for Petitioner
AO Kaspersky Lab*

Date: March 29, 2024

KING & SPALDING LLP

BY: /s/ Brian Eutermoser
Brian Eutermoser
Registration No. 64,058

Attorney for Patent Owner *Open Text Inc.*