UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT INC.
Patent Owner

Case No. IPR2023-00289
Patent No. 8,418,250

PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 8,418,250

TABLE OF CONTENTS

I. INTRODUCTION

CrowdStrike, Inc. ("Petitioner") respectfully requests *inter partes* review ("IPR") of claims 1-30 ("Challenged Claims") of U.S. Patent No. 8,418,250 ("'250 Patent").

II. SUMMARY OF THE '250 PATENT

A. Description of the '250 Patent

Aimed at systems and methods for classifying computer objects as malware, the '250 Patent purports to provide "very different" malware classification techniques from conventional "signature matching." *'250 Patent* (Ex. 1001), 1:4-3:18; *'250 File History* (Ex. 1002), 401. Conventional approaches employed human analysts to perform a deep analysis on new objects such that a database of signatures (e.g., a hash or checksum) is manually updated to identify known malware. *'250 Patent*, 1:18-2:64. This introduced both delay and risk. *Id*., 2:57-64.

The '250 Patent's techniques are "founded on receiving information about objects present on a plurality of remote computers at a base computer" which "allows analysis to be performed mapping the behaviour and attributes of objects known across the community in order to identify suspicious behaviour and to identify malware at an early stage." *250 File History*, 401.

Specifically, a base computer receives information about an object's behavior from one or more remote computers, and the base computer may classify the object as unsafe based on that behavior data. *'250 Patent*, 9:4-10, 12:14-42. If the base

computer does not initially classify the object as malware, then the system generates a "mask" based on the normal actions performed by the object that "indicates the particular types of behaviour to be expected from the object." *Id.*, 14:55-15:4. The system then monitors the object's behavior as it runs, and the object can be reclassified as malware "if the actual monitored behavior extends beyond that permitted by the mask." *Id*., 4:21-27, 14:55-15:4; *see also id*., 15:5-9.

B. Summary of the Prosecution History

The '250 Patent was filed on June 30, 2006, claiming 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.

The '250 Patent's application underwent several rejections during prosecution, including a rejection over U.S. Patent Publication No. 2003/0177394 ("*Dozortsev*") and U.S. Patent Publication No. 2002/0099952 ("*Lambert*"). *'250 File History*, 565-571. The Examiner found "Dozortsev does not explicitly teach generating a mask that defines acceptable behavior for the computer object, wherein only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask[,]" but found *Lambert* teaches this. *Id.*, 566, 569.

Applicant amended the independent claims to require "the received data to include information about the behaviour of the object, and the mask to be generated [automatically] based on normal behaviour of the object determined from the received data (including behaviour information)." *Id*., 584. Applicant argued in *Lambert* "[t]here is no teaching or suggestion of receiving behaviour information and using normal behaviour to define a mask" and because "**an administrator classifies the software[**,]…**[t]his is clearly manual, and not automatic generation of a mask**." *Id.*, 585-586 (citations omitted)[1].

The Examiner rejected the amended claims over *Dozortsev* and *Kester*, finding "Dozortsev does not explicitly teach receiving data about the behavior of the object running on the remote computer, automatically generating a mask in accordance with normal behavior of the object determined from the received data that defines acceptable behavior for the computer object, and only allowing the computer object to continue to run when the behavior of the computer object is permitted by the mask, and disallowing the computer object to run when the actual monitored behavior extends beyond that permitted by the mask." *Id.*, 611. Instead, these limitations were taught by *Kester's* disclosure of a "workstation management module monitoring the behavior of an application[,]" "wherein the behavior of the

[1] All emphasis added by Petitioner unless noted otherwise.

application is used to determine the access privileges for the application" and "the privileges are used to determine what actions are permitted." *Id.*

In response, Applicant argued, "in Kester the hash/policy table 204 for each workstation stores details of previously seen programs (categorized programs) and previously seen network access requests (expected network access behavior) and policies for those programs and requests that have been set by the Network Administrator/human reviewer[,]" but *Kester* does not teach "reclassifying an object as malware if its monitored behavior extends beyond that permitted by the mask" nor "classifying an object as malware at all." *'250 File History*, 633. According to Applicant, "[t]his is clearly different from the claims of the present invention" because the claimed process requires "a mask is <u>automatically</u> generated defining acceptable behavior for the object based on the normal behavior of the object. The <u>automatically</u> generated mask is used to automatically monitor the operation of the object and to reclassify the object as malware if the behavior extends beyond the mask. **A human decision is not necessary to generate the mask or monitor the operation**." *Id.* (underline emphasis in original).

The Examiner agreed and issued a Notice of Allowance, stating "applicant's arguments filed on December 23, 2011 are persuasive, as such the reasons for allowance are in all probability evident from the record." *Id.,* 803, 839.

As explained in detail below, *Kester* in view of *Honig* renders obvious automatically generating a mask and classifying/reclassifying an object as malware.

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):

> (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.

Although the Proposed Grounds herein rely on the same *Kester* reference considered during prosecution, the office never previously considered the instant Grounds which rely on *Kester* in view of *Honig*, a new reference neither cited nor

considered during prosecution. As described in the preceding section, *Kester* was advanced in an office action as a secondary reference for the purpose of teaching "automatically generating a mask[.]" Patent Owner ("PO") insisted *Kester* did not teach this automatic process, and the Examiner agreed. The Proposed Grounds do not ask the Board to reconsider that conclusion. Instead, this Petition relies on *Honig* for its automatic mask generation. *See* Section II.B (Summary of the Prosecution History); Ground 1, Element 1.5 (reliance on *Honig* for "automatic" mask generation); Ground 1, Elements 1.2, 1.3, 1.4, and 1.10 (similarly mapping *Kester* in view of *Honig*). Because the Proposed Grounds rely on *Kester* in a fundamentally different way than it was analyzed during prosecution and because they include teachings from *Honig* not considered during prosecution, the instant grounds are materially different and non-cumulative to the rejections evaluated during examination.

Moreover, the instant grounds are significantly different from the *Dozortsev*/*Kester* and *Dozortsev*/*Lambert* combinations used in prosecution—each of which failed to address any "automatic" generation of a mask as disclosed by *Honig*. 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. Ex. 1007 (Dkt. 98, PO Opposition Brief), 4-22. For the purposes of this proceeding only, Petitioner proposes the Board adopt PO's proposal, with the exception of the term "automatically" for the reasons below. *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]

1. automatically

The term "automatically" is claimed in independent claims 1, 13 and 25. *See, e.g.*, Claim 1.5 and 1.7; Claim 13.4 and 13.5; Claim 25.4 and 25.6. As discussed in Section II.B, the file history is unequivocal that the term "automatically" excludes human involvement. *Supra*, Sec. II.B (prior to notice of allowance, distinguishing claims from prior art because in the claimed "automatic[]" process, "[a] human decision is not necessary to generate the mask or monitor the operation"); *see also id.*, 585-586 (distinguishing *Lambert* because "an administrator classifies the software[,]…[t]his is clearly manual, and not automatic generation of a mask"). In litigation, however, PO expressly interprets "automatically" to encompass "human involvement." Ex. 1007 (Dkt. 98 *PO Opposition Brief*), 9-10 ("The prosecution history does not require 'automatically' to be defined as 'without human involvement'…it is a bridge too far to say that automatically must be without ***any*** human involvement at all") (emphasis in original). PO's position conflicts with clear and unambiguous statements made during prosecution.

[2] Petitioner reserves the right in other proceedings to construe the Challenged Claims and pursue indefiniteness theories.

As noted above, it is the Board's practice to accept a PO's interpretation for purposes of *Inter Partes* Review. Petitioner acknowledges, however, that the Board is not obligated to accept a facially overbroad interpretation that directly conflicts with the prosecution history. Accordingly, Petitioner asks the Board apply PO's interpretation of the claims as allowing human involvement except where the claims recite "automatically," which the Petition assumes requires a step performed "without human involvement."

E. Level of Skill of a Person Having 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*.") (Ex. 1003), ¶¶38-41.

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

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

Petitioner certifies the '250 Patent is eligible for IPR.

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

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

Proposed Ground of Unpatentability	Exhibits
Ground 1: Claims 1-2, 7-14, and 19-30 are **obvious** under §103(a) over U.S. Patent Publication No. 2005/0210035 to Kester et al. ("*Kester*") in view of U.S. Patent No. 7,225,343 to Honig et al. ("*Honig*")	Ex. 1004 Ex. 1005
Ground 2: Claims 3-6 and 15-18 are **obvious** under §103(a) over *Kester* in view of *Honig* and U.S. Patent No. 7,594,272 to Kennedy et al. ("*Kennedy*")	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* (Ex. 1004).

According to the '250 Patent, "the present invention relates to a method and apparatus for classifying a computer object as malware." *'250 Patent*, 1:4-7; *see also id.*, Claims 1, 13. *Kester* also provides a method and apparatus for classifying a computer object as malware. *Kester*, [0134], [0098] (discussing "classifying applications" into at least one parent group and/or category), Fig. 4A (including "Malware" parent group and "Malicious Applets and Scripts" category); *see also*

Ground 1, Elements 1.2 and 1.3. Accordingly, *Kester* is in the same field of endeavor as the ’250 Patent. *Decl.*, ¶¶57-63.

Kester is also reasonably pertinent to certain problems addressed by the ’250 Patent such as the need to initially classify an object as not malware, to generate a mask for the object that defines acceptable behavior for the object, and to reclassify the object if the actual monitored behavior is not normal (extending beyond the mask). *Compare ’250 Patent,* 4:21-30, 15:2-4, *with Kester*, [0076] (describing an application that is “not operating in a predetermined manner [is] categorized or re-categorized”), [0179]-[0181], Claims 1-2; *Decl.*, ¶¶57-63. Therefore, *Kester* is analogous art to the ’250 Patent.

B. 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* (Ex. 1005).

Like the ’250 Patent, *Honig* teaches a method and system for classifying applications as normal or anomalous. *Compare Honig*, 15:43-16:24, *with ’250 Patent*, 1:9-12. Accordingly, *Honig* is in the same field of endeavor as the ’250 Patent. *Decl.*, ¶¶64-66.

Honig is also reasonably pertinent to certain problems addressed by the ’250 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 is not normal (extending beyond the mask). *Compare '250 Patent*, 4:21-30, 15:2-4, Claims 1, 13, 25, *with 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 ("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…model generation…and to solve many of the practical problems associated with the deployment of data mining-based IDSs."); *Decl.*, ¶¶64-66. Therefore, *Honig* is analogous art to the '250 Patent.

C. Kennedy

Kennedy was filed on October 5, 2004, issued on September 22, 2009, and qualifies as prior art to the '250 Patent under 35 U.S.C. §§102(e) (pre-AIA). *Kennedy* (Ex. 1006).

Like the '250 Patent, *Kennedy* teaches a malicious software detection module that includes a file set tracking and behavior module for declaring software malicious. *Compare Kennedy*, 1:6-9; *see also id*., Abstract, Fig. 4 (step 418), *with '250 Patent*, 1:9-12. Accordingly, *Kennedy* is in the same field of endeavor as the '250 Patent. *Decl.*, ¶¶67-69.

Kennedy is also reasonably pertinent to certain problems addressed by the '250 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 '250 Patent*, 2:57-3:18, 3:31-41, *with Kennedy*, 1:11-52; *Decl.*, ¶¶67-69. Therefore, *Kennedy* is analogous art to the '250 Patent.

V. THERE IS A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. Ground 1: *Kester* in view of *Honig* Renders Claims 1-2, 7-14, and 19-30 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 *Honig* renders obvious a method of classifying a computer object as malware. *See* Elements 1.2 and 1.3 below.

[1.1] at a base computer, receiving data about a computer object from each of plural remote computers on which the object or similar objects[3] are stored, the

[3] Petitioner proposes the term "similar objects" is indefinite in litigation. Ex. 1010 (Defendants' Opening Markman Brief), 2-4. Regardless of the indefiniteness of this term, the claims are still rendered obvious by art cited herein. Because the claim is written in the alternative, any uncertainty as to the outer boundaries of "similar objects" does not preclude application of prior art that discloses "receiving data

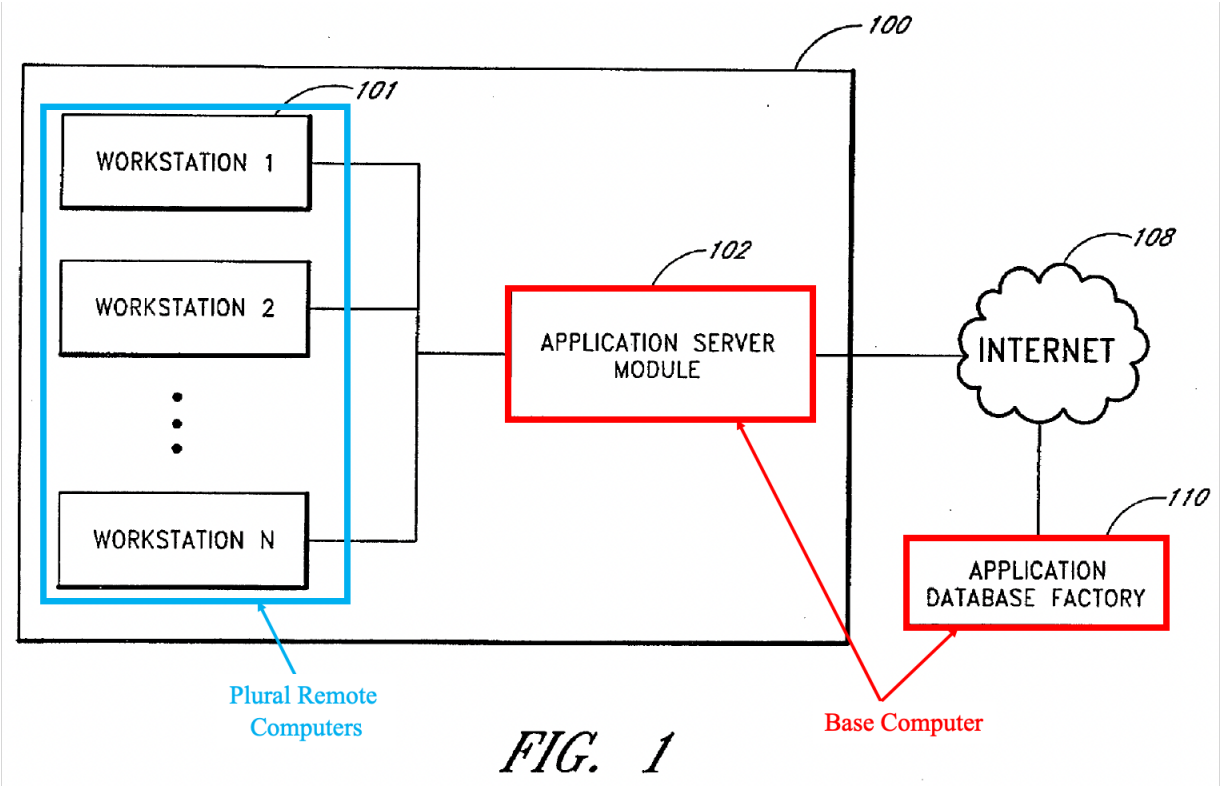
data including information about the behaviour of the object running on one or more remote computers;

Kester teaches a *base computer* (application server module 102 and/or application database factory 110) that receives data about a *computer object* (program/application) from each of *plural remote computers* (workstations 101) on which the object is stored. *Kester*, [0008], [0036], 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 '250 Patent. *'250 Patent*, 1:14-17 (object includes "any executable computer file"), 7:63-66 (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"). And this same application can be stored on the plurality of workstations 101 (i.e., plural remote computers). *Kester*, [0041], [0157] ("different workstations 101 can have…the same application running thereon").

---

about a computer object from each of plural remote computers on which **the object** [is] stored[.]"

Kester's architecture is shown below:



Kester, Fig. 1 (annotated).

The received data (application related data) includes information about the behavior of the application—namely, "network access data." For example, when an application on a workstation is launched, 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," and notes it may include any information associated with the requested application such as behavior data (e.g., network access data) and other information about the

application (e.g., application name, hash, publisher, suite, file size, directory location, execution request frequency). *Kester*, [0061]-[0062], [0073], [0051], [0099], [0141]. Exemplary "network access data" (NET_ACTIVITY_INFO) is depicted in Fig. 4B as including information about the application and its behavior (transport protocol, destination port, and destination IP address). *Kester*, Fig. 4B, [0016], *see also id*., [0083].

The application server module and/or application database factory[4] (*base computer*) receives the collection data from each remote workstation. Namely, each workstation uploads collection 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. This collection data can be uploaded from multiple (potentially thousands) of workstations. *Kester*, [0119]. This process is shown below:

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

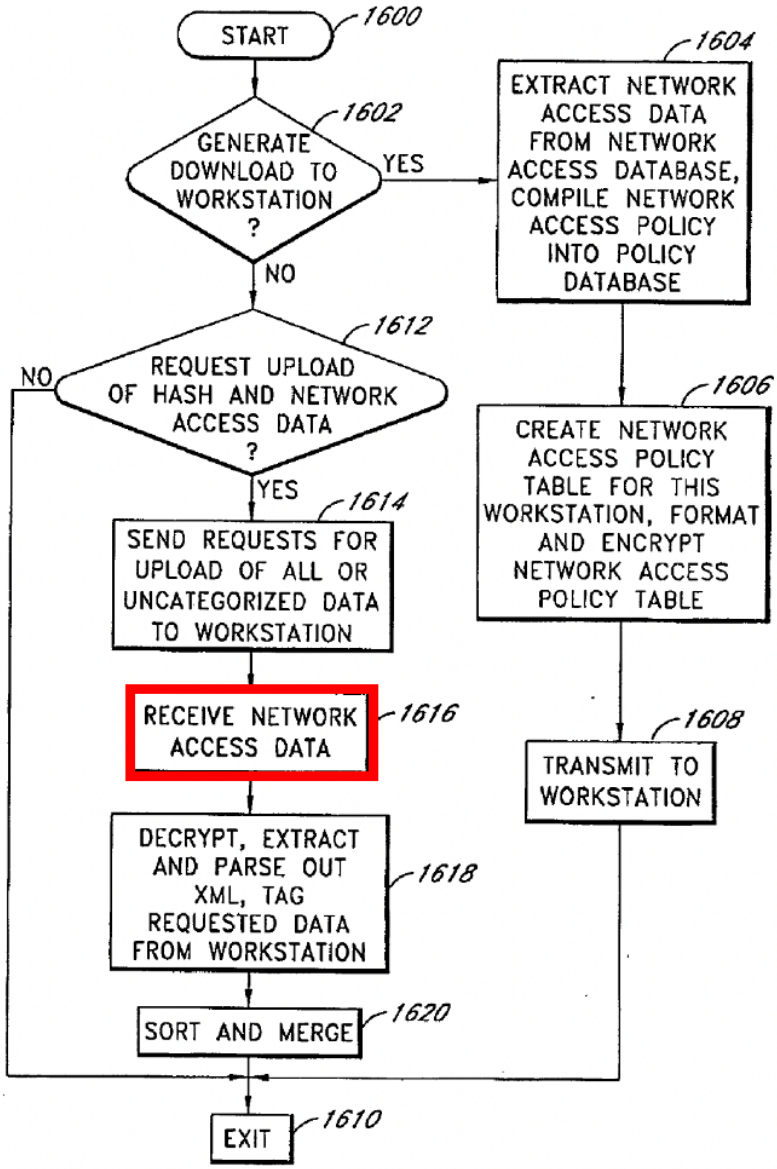


FIG. 16

Kester, Fig. 16 (annotated).

The application server module 102 may then transmit collection data to the application database factory 110 "immediate[ly] or periodic[ally]." *Kester*, [0078], Fig. 3; *see generally id.*, [0086]-[0087], [0132], [0176], Figs. 11, 12, 18, 19.

Exemplary collection data received by and displayed at the application database factory is illustrated in Figure 6. *Kester*, [0097]-[0098], Fig. 6.

Thus, *Kester* renders obvious Element 1.1. *Decl.*, ¶¶71-79.

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

Kester teaches determining in the application server module 102 and/or application factory database 110 (*base computer*) whether the collection data (*data about the computer object received from plural computers*, *see* Element 1.1) indicates the application is malware.

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 application category is "malicious applets and scripts"—i.e., malware. *See '250 Patent*, 1:14-17 (defining "malware"); *Decl.*, ¶¶80-82. An exemplary "malware" parent group and a "malicious applets and scripts" application 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 application behavior data is malware or not, *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," (*see Kester*, [0071) by analyzing "each application and any additional data that is associated with the application to determine its appropriate category or categories." *Kester*, [0134], *see also id.*, [0071], [0078], [0084], [0122], Figs. 3, 10; *see also id*., Fig. 20 (describing categorization process at application database factory 210).

Because *Kester* teaches determining 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* determines whether this data indicates the application is malware. *Decl.*, ¶¶83-88.

Alternatively,[5] *Kester* in view of *Honig* renders Element 1.2 obvious. Namely, it would have been obvious to implement *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

[5] During prosecution, Applicant indicated an intent to exclude human involvement by amending the claims to require a step is performed "automatically." *Supra*, Section II.B. These "automatic" processes are claimed in Elements 1.5 and 1.7 below. In litigation, PO argued that human intervention is not excluded even for limitations that recite an "automat[ed]" process. *Supra*, Section II.D.1. Under PO's logic, if not even the limitations that recite "automatically" exclude human involvement, then no limitations exclude human involvement. Accordingly, this Petition applies PO's view that no limitations exclude human involvement, apart from Elements 1.5 and 1.7. For example, the requirement in Element 1.2 that a determination takes place "in the base computer" is satisfied where a human uses the base computer to make the requisite determination. Should PO reverse course here to argue other elements of claim 1 foreclose human involvement, *Kester* in view of *Honig* describes a computer-implemented method of classifying a computer object as malware for the reasons herein.

determine whether received data indicates the application is malware, as taught by *Honig*. *Decl.*, ¶¶89-98.

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, 21:39-24:31; *Decl.*, ¶93. *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.*, ¶94.

It would have been obvious to implement *Kester's* application server module and/or database factory to determine whether received data indicates a program/application is malware as taught by *Honig*. *Decl.*, ¶¶95-98. 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 validating and/or modifying the automated determination, and therefore would not fundamentally change the overall operation of *Kester's* method. *Decl.*, ¶¶95-98. A PHOSITA would have been motivated to implement the combined *Kester/Honig* method in this way to reduce 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.*, ¶¶95-98.

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 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*.

Thus, *Kester* or *Kester* in view of *Honig* renders obvious Element 1.2. *Decl.*, ¶¶80-98.

[1.3] classifying the computer object as malware when the data indicates that the computer object is malware;

Kester or alternatively *Kester* in view of *Honig* teaches determining an application to be malware when the collection data so indicates and assigning an appropriate category/parent group to this application (e.g., "malicious applets and scripts" or "malware") thus classifying this object as malware. *See* Element 1.2.

More specifically, *Kester* teaches using a classification user interface 106 and/or application analyst classification module 302 to classify the application into one or more "appropriate" categories. *See* Element 1.2; *Kester*, [0092], Claim 2; *Decl.*, ¶¶99-103. Regarding module 302, *Kester* discloses this module may "display[] this information to [a] human reviewer to aid in categorizing the application." *Kester*, [0097]. An exemplary display is shown below:

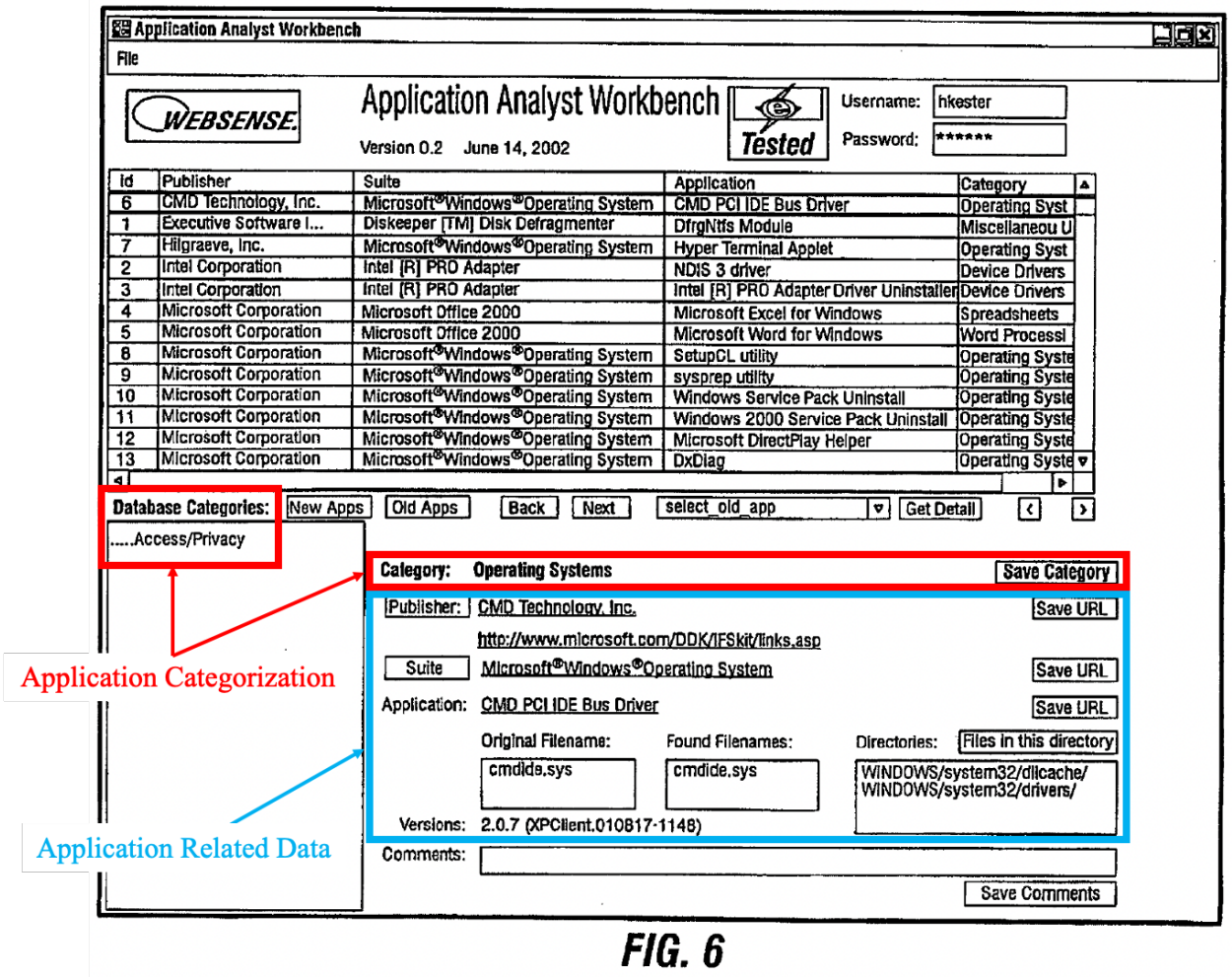


FIG. 6

Kester, Fig. 6 (annotated). This application could be classified into a category/parent group, such as "malware" and "malicious applets and scripts." *Kester*, [0082], [0098], Fig. 4A. A PHOSITA would have understood *Kester* teaches when the application related data indicates an application is a malicious applet or script, or other type of malware, it is classified into the respective appropriate category. *Decl.*, ¶¶99-103.

Alternatively, it would have been obvious to implement *Kester's* application classification process, such that the application server module and/or database

factory initially classify and/or reclassify applications into the appropriate group/category (e.g., malware) without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above. *See* Element 1.2; *Decl.*, ¶¶104-107.

Thus, *Kester* or *Kester* in view of *Honig* renders obvious Element 1.3. *Decl.*, ¶¶99-107.

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

Kester teaches initially classifying the program/application as something other than malware (e.g., Access/Privacy) when the Element 1.2 determination indicates the program/application is not malware and, thus, discloses Element 1.4.

Kester teaches classifying the application into one or more "appropriate" categories. *See* Element 1.3. 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.*, ¶¶108-112. 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.

As discussed in more detail in Element 1.10, *Kester* teaches "[a] human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization[,]" meaning an application can be initially classified into one category/group (e.g., operating systems or access/privacy) and subsequently recategorized into a different category/group (e.g., malicious applets and scripts or malware). *Kester*, [0089], [0071] (describing same process for module 102); *Decl.*, ¶¶108-112.

Alternatively, it would have been obvious to implement *Kester's* application classification process, such that the application server module and/or database factory initially classify and/or reclassify applications into certain parent groups/categories (including non-malware groups and categories) without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above. *See* Element 1.2; *Decl.*, ¶¶113-114.

Thus, *Kester* or *Kester* in view of *Honig* renders obvious Element 1.4. *Decl.*, ¶¶108-114.

[1.5] automatically generating a mask for the computer object that defines acceptable behaviour for the computer object, wherein the mask is generated in accordance with normal behaviour of the object determined from said received data;

In litigation, PO interprets the plain meaning of these "mask" limitations to "encompass[] any suitable way to define acceptable behavior" for an object. Ex. 1007 (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.1.

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"). Although *Kester* does not explicitly state the mask is generated "automatically," such an implementation would have been obvious based on the teachings of *Kester* in view of *Honig*. *Supra*, Section II.D.1 (construing "automatically" as "without human involvement"); *Decl*., ¶¶115-132.

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 (*normal*) 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*, [0083], Fig. 4B.

Like *Kester*, the '250 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. *'250 Patent,* 14:55-58, 15:2-4. Accordingly, a PHOSITA would have understood *Kester* teaches generating a mask for the computer object that defines acceptable behavior for the object, wherein the mask is generated in accordance with normal behavior of the object determined from said received data. *Decl.*, ¶¶117-120.

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*., ¶¶121-132. As described above, *Honig* teaches a known process for generating adaptive learning models to inform malware determinations/classifications. *See* Elements 1.2 and 1.3.

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 determination/classification. *Decl.*, ¶¶123-124.

It would have been obvious to "automatically" generate (e.g., without human input) *Kester's* mask in view of the teachings of *Honig*. *Decl.*, ¶¶125-131. 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.*, ¶¶125-131. 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.*, ¶¶121-131. 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*, 7:38-41, 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, 21:39-24:31; *Decl.*, ¶¶121-131. 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"* (Ex. 1019); *Decl.*, ¶129. 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.*, ¶¶121-131. 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, thereby improving similar malware detection method/systems in the same way. *Id*.; *Kester*, [0090], [0135] ("hints" enhance productivity of human reviewer for classification).

Thus, *Kester* in view of *Honig* renders Element 1.5 obvious. *Decl.*, ¶¶115-132.

[1.6] running said object on at least one of the remote computers;

Kester discloses running the same program/application (*said object*) on one or more remote workstations. In *Kester*, the program/application runs on a remote computer, which allows the *Kester* system to "monitor the ongoing network activity or behavior of the application." *Kester*, [0041], [0157] ("different workstations 101 can have different policies associated with the application running thereon"); *see generally id.*, [0137], [0049]-[0055], [0057], [0059]-[0060], [0099]-[0104], Abstract, Figs. 2, 7; *see also* element 1.7.

Thus, *Kester* renders obvious Element 1.6. *Decl.*, ¶¶133-134.

[1.7] automatically monitoring operation of the object on the at least one of the remote computers;

Kester teaches a network access detection module 208 on each remote workstation that continually and automatically monitors the operation of the program/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.

As noted by PO during prosecution, "automatically monitor[ing]" at least includes "not monitor[ing] by a human operator." *'250 File History*, 447. As above, *Kester* discloses this. *See also Decl.*, ¶¶135-136.

Thus, *Kester* renders obvious Element 1.7. *Decl.*, ¶¶135-137.

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

Kester teaches "allowing the application to access the network" (*allowing the computer object to continue to run*) when the network accessing application properties/attributes match "with an expected behavior for the application" (*behavior of the object is permitted by the mask*). *Kester*, [0055], [0057], [0143]. Namely, "[w]hen an application or program accesses a network, the network access detection module [208] monitors the behavior or activity of the application or program." *Kester*, [0051].

First, the application digest generator 201 "generates a digest of data relating to the application" including properties/attributes such as IP address(es), port(s), protocol, and/or other network access data (also referred to as "collection data"). *Kester*, [0141]; [0051]; Fig. 14 (step 1404).

Second, the network access detection module 208 of the workstation 101 "compares the application digest and collection data prepared by the application digest generator 201 to the hash/policy table 204." *Kester*, [0142], [0053], [0055], Fig. 14 (step 1406).

Third, when there is a match (i.e., "the behavior or network attributes of the application matches with an expected behavior for the application"), the associated one or more access privileges/policies/rules (e.g., "allowing execution of the program" and/or "allowing the application to access the network") are applied.

Kester, [0143]; [0055], [0057], Fig. 14 (step 1410). *See generally Kester*, [0042], [0050]-[0057], [0072], [0137]-[0145]. This is shown below in annotated Fig. 14:

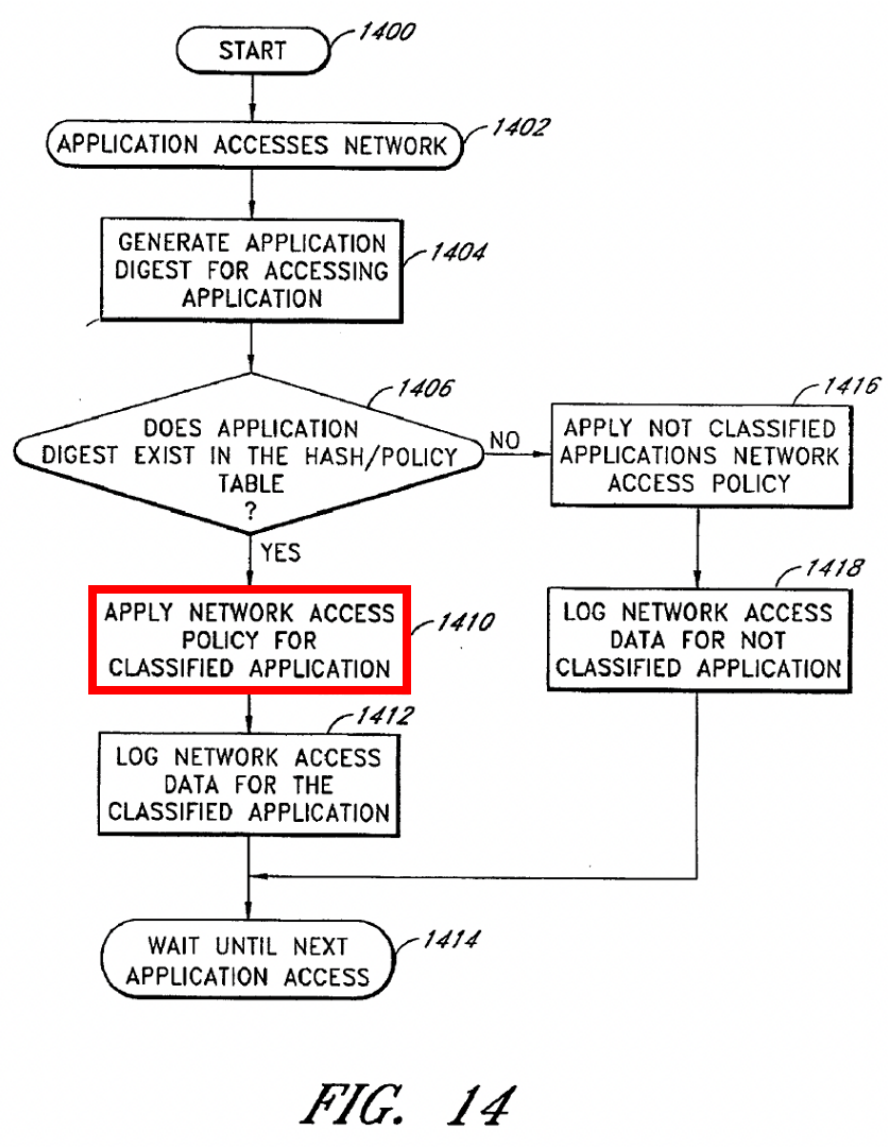


FIG. 14

Kester, Fig. 14 (annotated).

Thus, *Kester* renders obvious Element 1.8. *Decl.*, ¶¶138-140.

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

Kester teaches "denying access to the network" (*disallowing the computer object to run*) when the network accessing application properties/attributes (*the actual monitored behavior of the computer object*) do not match an expected behavior for the application (*extends beyond that permitted by the mask*). *Kester*, [0055], [0059], [0145].

For example, expected network activity (as listed in the hash/policy table 204 of each remote workstation) includes "attributes [that] are associated with the application when the application accesses the network in an expected manner" (e.g., a specific protocol, a specific IP address, and a specific access port). *Kester*, [0184]. According to *Kester*, "[i]f the application requests access to the network using a different protocol than the expected protocol listed in the hash/policy table 204, the network access detection module 208 may disallow access." *Id.*

The process of disallowing a computer object to run when the actual monitored behavior extends beyond that permitted by the mask includes first generating application digest and second comparing digest and collection data to hash/policy table 204. *See* Element 1.8 for detailed discussion.

Third, when "the application digest and collection data does not correspond with an application or hash classified in the hash/policy table 204," this means "the application is behaving unexpectedly." *Kester*, [0145], [0059]; *Decl.*, ¶¶141-143. Here, "the network access detection module 208 applies a not-classified application

policy to the request to access to the network." *Kester*, [0145]; Fig. 14 (step 1410). According to *Kester*, the not-classified application policy/rule can include "stop[ping] execution or network access for the application[,]" "denying execution of the program," and/or "denying access to the network[.]" *Kester*, [0055], [0059], [0145]. *See generally Kester*, [0050]-[0059], [0137]-[0145], [0184]. This is shown below in annotated Fig. 14:

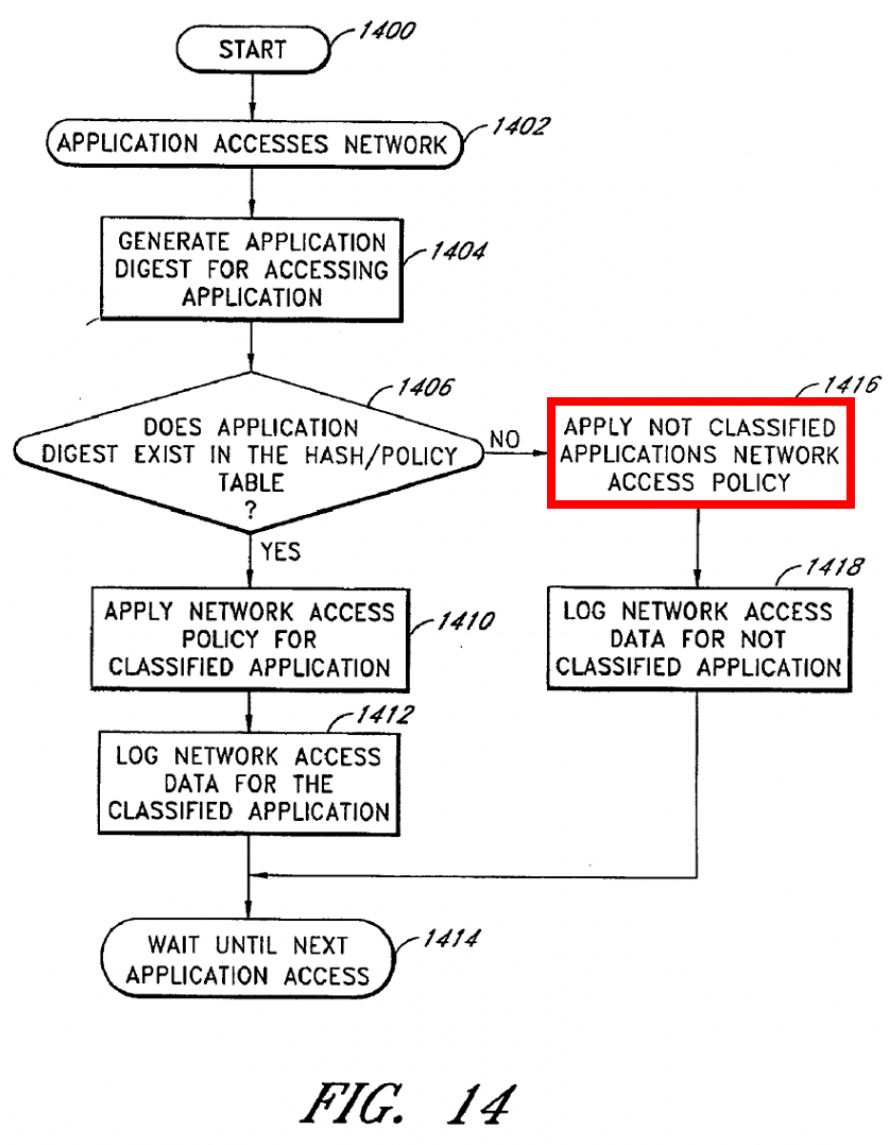


FIG. 14

Kester, Fig. 14 (annotated).

Thus, *Kester* renders obvious Element 1.9. *Decl.*, ¶¶141-144.

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

Kester or *Kester* in view of *Honig* teaches re-categorizing (reclassifying) the program/application as malware when the program/application is not operating in an expected manner as set forth below.

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], [0071], [0074], [0084], [0151], [0153]. 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 in Elements 1.2 and 1.3. *Kester*, [0089]-[0094]; *Decl.*, ¶¶145-147. *Kester* teaches reclassifying the program/application as malware when the collection data so indicates. *See* Elements 1.2, 1.3.

Alternatively, it would have been obvious to implement *Kester's* application classification process, such that the application server module and/or database factory initially classify and/or reclassify applications into the malware parent group/category without human intervention using an adaptive learning model as taught by *Honig* for the reasons described above. *See* Element 1.2; *Decl.*, ¶149.

Thus, *Kester* or *Kester* in view of *Honig* renders obvious Element 1.10. *Decl.*, ¶¶145-149.

2. Claim 2

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

within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

Kester teaches the collection data sent from the remote workstations to the base station may include "application name" (*current object name*), "original Filename" (*original object name*), "directory location" (*physical and folder location on disk*), "file size" (*size of the object*), "publisher, suite,…version" (*vendor, product and version*), and network access request "time of day, port, I.P. address, protocol" (*events initiated by or involving the object when the object runs on the respective remote computers*). *Kester*, [0061]-[0062], [0051], [0119], [0097], Fig. 6; *see also* Element 1.1; *Decl.*, ¶¶150-151. Additionally, this collection data may include a hash generated from executable instructions contained within or constituted by the object. *See* Claim 8 below (describing "hash for the application is determined by transforming the binary associated with the application into a unique set of bits[,]" *see Kester*, [0052]).

3. Claim 7

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

Kester teaches the collection data (*see* 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). In

embodiments, "the hash value[] includes network access data[.]" *Kester*, [0011], Claim 39; *Decl.*, ¶¶152-154.

4. Claim 8

8. A method according to claim 7, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

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.*, ¶¶155-156. This is consistent with the '250 Patent's disclosure of an MD5 Type 1 checksum (*key*). *'250 Patent*, 2:13-17, 9:63-10:32; *Decl.*, ¶¶155-156.

5. Claim 9

9. A method according to claim 7, wherein the key has at least one component that represents data about said object.

Kester's hash (*key*) includes network access data. *See* Claim 7; *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.*, ¶157. Additionally or alternatively, network access data includes the "name, publisher, suite, [] file size, [and/or] version" of the network accessing application. *Kester*, [0051]; *Decl.*, ¶157. Thus, *Kester's* hash has at least one component that represents data about the object. *Decl.*, ¶¶157-158.

6. Claim 10

10. A method according to claim 9, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

See Claim 9.

7. Claim 11

11. A method according to claim 7, wherein the key has at least one component that represents the physical size of the object.

Kester's hash (*key*) includes network access data (*see* Claim 7), which includes the network accessing application's "file size[.]" *Kester*, [0051], [0011], Claim 39. Thus, *Kester's* hash has at least one component that represents the physical size of the object. *Decl.*, ¶¶160-161. *See also* Claims 7, 9.

8. Claim 12

12. A method according to claim 1, comprising if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, comparing at the base computer the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects, allowing said behaviour for the object and modifying the mask to allow that behaviour for that object in future.

As discussed above, Petitioner understands PO has interpreted the claims in litigation, including claim 12, to encompass human involvement. *Supra*, Sections II.B, II.D.1; Claim 1.2, Fn. 5. Petitioner provides its mapping below pursuant to PO's purported interpretation of the claims.

Kester teaches when an application is behaving unexpectedly (*beyond that permitted by the mask*), it is considered an "uncategorized application" and its collection data (including the unexpected behavior) is transmitted to the server module and/or database factory (*base station*) for categorization/re-categorization. *Kester*, [0070]-[0071], [0075]-[0076], [0059], [0084], [0087], [0151]-[0153]; *see* Element 1.10.

Kester teaches an application "not operating in a predetermined manner [is] categorized or re-categorized" and that "[t]he application is re-categorized based on parsed properties or features which are different than the parsed properties or features associated with the original categorization of the application." *Kester*, [0076]. Thus, *Kester* teaches comparing at the base computer the non-permitted behaviour (e.g., "on parsed properties or features which are different") with the

behaviour of other objects (e.g., "the parsed properties or features associated with the original categorization of the application"). *Decl.*, ¶¶162-167; *see also Kester* [0044], [0182] ("expected network activity" used to inform a categorization "can be determined from network activity by a different but related application").

When an application is behaving unexpectedly, "the network access detection module 208 [of the remote workstation] applies a not-classified application policy to the request to access the network." *Kester*, [0145], [0059], Fig. 14 (step 1410). According to *Kester*, the not-classified application policy/rule can include "allowing the application to access the network." *Kester*, [0042], [0055], [0059], [0072], [0145]. *See generally Kester*, [0050]-[0059], [0137]-[0145], [0184], Fig. 14.

Moreover, categorization and re-categorization in *Kester* can include categorizing/re-categorizing the expected or allowed network activity for the application (*modifying the mask*). *Kester*, [0089] ("A human reviewer interacts with the application analyst's classification module 302 to perform the categorization or recategorization. The process for classifying or categorizing applications at the application database factory is described with reference to FIGS. 13 and 20."), *Kester*, [0178]-[0187], Fig. 20 (step 2002, where the list of uncategorized applications is extracted for classification and step 2008, "determine allowed behavior for each application"); *see also id.*, [0071], Fig. 17. A PHOSITA would have understood *Kester* discloses re-categorizing the expected or allowed network

activity for an application and such re-categorization may be based on parsed properties or features which are different than the parsed properties or features associated with the original categorization of the application. *Decl.*, ¶¶162-167.

Regarding categorizing and/or re-categorizing, *Kester* contemplates that a human reviewer may determine whether behavior is allowed, including by analyzing network activity from multiple other workstations and performing additional research. *See, e.g., Kester*, [0178]-[0187], [0163]-[0164], Figs. 17, 20. A PHOSITA would have understood *Kester's* broad disclosures encompass a situation where the unexpected behavior is known to be not malicious for the same application operating on other workstations to which it is compared and re-categorizing (*modifying*) the expected or allowed network activity for the application (*mask*) to allow the behavior in the future. *Decl.*, ¶¶162-167.

Alternatively, *Kester* in view of *Honig* renders claim 12 obvious. *Decl*., ¶¶168-171. First, *Honig* teaches a model generator 16 that automatically generates a model of behavior for a computer object in accordance with normal behavior. *See* Claim 1, Element 1.5. *Honig* further teaches that models "become out of [the] data" and are "update[ed] and alter[ed] [] on the fly." *Honig*, 19:46-62. Updated models are automatically distributed so that detectors have the "most recent" model for properly classifying behavior as normal or malicious. *Honig*, 19:46-62, 14:42-51, 14:27-31, 8:14-24.

It would have been obvious to configure *Kester's* server module and/or database factory user interface to alert the human reviewer when unexpected behavior occurs and allow the user to indicate the unexpected behavior is known to be not malicious and modify the mask to allow the behavior for the application in the future based on the teachings of *Kester* and *Honig*. *Decl.*, ¶¶168-171. A PHOSITA would have recognized this implementation to be an obvious and predictable improvement to *Kester's* application file monitoring system/method. *Id*. As noted above, *Kester's* disclosures already contemplate such an implementation, but do not detail how this implementation would work. A PHOSITA would have been motivated to apply the teachings of *Honig* to ensure when unexpected behavior actually represents normal operation, *Kester's* expected network activity is modified accordingly, thereby reducing false alarms and nuisance to *Kester's* employee end users. *Id*. This implementation 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*. Given the triviality and prevalence of providing such alerts and prompting for user input, there would have been a reasonable expectation of success configuring the combined method/system in this way. *Id*.

Thus, *Kester* or *Kester* in view of *Honig* renders obvious Claim 12. *Decl.*, ¶¶162-172.

9. Claim 13

13[P]. 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* Elements 1.2 and 1.3 above.

[13.1] a base computer constructed and arranged to receive data about a computer object from each of plural remote computers on which the object or similar objects are stored, the data including information about the behaviour of the object running on one or more remote computers;

See Claim 1, Element 1.1.

[13.2] the base computer being constructed and arranged to determine whether the data about the computer object received from said plural computers indicates that the computer object is malware; and,

See Claim 1, Element 1.2.

[13.3] the base computer being constructed and arranged to classify the computer object as malware when the base computer determines that the data indicates that the computer object is malware,

See Claim 1, Element 1.3.

[13.4] wherein the base computer is constructed and arranged to automatically generate a mask for an object that is initially classed not as malware, said mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data,

See Claim 1, Elements 1.4 and 1.5.

[13.5] wherein operation of the object is automatically monitored when the object is running on at least one of the remote computers,

See Claim 1, Element 1.6 and 1.7.

[13.6] the object being allowed to continue to run when behaviour of the object is permitted by the mask,

See Claim 1, Element 1.8.

[13.7] the object not being permitted to run when the actual monitored behaviour of the object extends beyond that permitted by the mask, and

See Claim 1, Element 1.9.

[13.8] the object is reclassified as malware when the actual monitored behaviour extends beyond that permitted by the mask.

See Claim 1, Element 1.10.

10. Claim 14

14. Apparatus according to claim 13, the data includes one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

See Claim 2.

11. Claim 19

19. Apparatus according to claim 13, wherein the base computer is constructed and arranged so as to be able to process data that is sent in the form of key that is obtained by a hashing process carried out in respect of the objects on said respective remote computers.

See Claim 7.

12. Claim 20

20. Apparatus according to claim 19, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

See Claim 8.

13. Claim 21

21. Apparatus according to claim 19, wherein the key has at least one component that represents data about said object.

See Claim 9.

14. Claim 22

22. Apparatus according to claim 21, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

See Claim 10.

15. Claim 23

23. Apparatus according to claim 19, wherein the key has at least one component that represents the physical size of the object.

See Claim 11.

16. Claim 24

24. A method according to claim 13, wherein, if the monitored behaviour of the object running on the remote computer exhibits non-permitted behaviour that extends beyond that permitted by the mask, the base computer is arranged to compare the non-permitted behaviour with the behaviour of other objects and if said behaviour is known to be not malicious for said other objects the base

computer is arranged to allow said behaviour for the object and to modify the mask to allow that behaviour for that object in future.

See Claim 12.

17. Claim 25

25[P]. A method of providing data about a computer object from a remote computer to a base computer so that a comparison can be made at the base computer with similar data received from other remote computers, the method comprising:

To the extent the preamble is limiting, *Kester* teaches a method of providing data about a computer object from a remote computer to a base computer. *See* Claim 1, Element 1.1 above. *Kester* further teaches such data is provided so a comparison can be made at the base computer (application server module 102 and/or application database factory 110) with "similar data" received from other remote computers. *See, e.g., Kester*, [0069] ("The network access data is uploaded to the application server module 102…can be compared to expected network access data for the application…[and] can be a compilation of network access data associated with contemporaneous or prior network access by the application…[or] can be compiled from the requesting workstation or from other workstations"); *see also id.*, [0076], [0179]-[0187], Figs. 10, 13, 17, 20. *See also* Claim 1, Element 1.5; *Decl.*, ¶¶189-190.

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

See Claim 1, Element 1.1.

[25.2] the data including information about the behaviour of the object running on one or more remote computers and including one or more of: executable instructions contained within or constituted by the object; the size of the object; the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers;

See Claim 1, Element 1.1 and Claim 2.

[25.3] the data being sent in the form of key that is obtained by a hashing process carried out in respect of the object on the remote computer,

See Claims 7-11.

[25.4] automatically generating a mask for the computer object, the mask defining acceptable behaviour for the object built in accordance with normal behaviour of the object determined from said received data;

See Claim 1, Element 1.5.

[25.5] executing the computer object at the remote computer;

See Claim 1, Element 1.6.

[25.6] automatically monitoring operation of the computer object at the remote computer compared to the behaviour permitted by the mask;

Kester teaches a network access detection module 208 on each remote workstation that continually and automatically monitors the operation of the program/application (*the object*). *See* Claim 1, Element 1.7. The network access detection module 208 in so doing, "a hash generated by the application digest generator 201 and collection data can be compared to hashes from the hash/policy

table 204 and network access data associated with the hash" where the "network access data associated with the hash" is the "expected behavior for the application" (*behavior permitted by the mask*). *Kester*, [0142]-[0143]; *see also id.*, [0053], [0055], Fig. 14 (step 1406); Claim 1, Element 1.5 (describing mask), Element 1.7 (describing *Kester's* automatic monitoring process), Element 1.8 (describing behaviors permitted by the mask).

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

See Claim 1, Element 1.8.

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

See Claim 1, Element 1.9.

[25.9] reclassifying the object as malware when the actual monitored behaviour extends beyond hat [sic] permitted by the mask,

See Claim 1, Element 1.10.

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

See Claim 1, Element 1.1.

[25.11] wherein the step of generating a mask for that object comprises building the mask with the base computer in accordance with normal behaviour of the object determined from said received information.

See Claim 1, Element 1.5.

18. Claim 26

26. A method according to claim 25, wherein the key has at least one component that represents executable instructions contained within or constituted by the object.

See Claim 8.

19. Claim 27

27. A method according to claim 25, wherein the key has at least one component that represents data about said object.

See Claim 9.

20. Claim 28

28. A method according to claim 27, wherein said data about said object includes at least one of: the current name of the object; the physical and folder location of the object on disk; the original name of the object; the creation and modification dates of the object; vendor, product and version and any other information stored within the object; the object header or header held by the remote computer; and, events initiated by or involving the object when the object is created, configured or runs on the respective remote computers.

See Claim 10.

21. Claim 29

29. A method according to claim 25, wherein the key has at least one component that represents the physical size of the object.

See Claim 11.

22. Claim 30

30. A non-transitory storage medium storing a computer program comprising program instructions for causing a computer to perform the method of claim 25.

Kester teaches a *non-transitory storage medium* (e.g., "addressable storage medium") storing a *computer program* (e.g., software) comprising *program*

instructions for causing a computer to perform the method steps of claim 25. *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 components or modules may be advantageously configured to reside on the addressable storage medium and configured to execute on one or more processors"); *see* Claim 25; *Decl.*, ¶206.

B. Ground 2: *Kester* in view of *Honig* and *Kennedy* Renders Claims 3-6 and 15-18 Obvious

1. Claim 3

3. A method according to claim 1, wherein determining identifies relationships between the object and other objects.

In *Kester*, the application server module 102 and/or application database factory 110 receives application related data "and additional information associated with the requested application" (i.e., collection data) to determine a group/category for the application, such as "malware." *See,* Claim 1, Elements 1.1, 1.2, 1.3. *Kester* also teaches "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 identifying "different but related objects" in the base computer. *Decl.*, ¶207.

Kester does not detail how it determines an application is "different but related," nor 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. *Kennedy* teaches this.

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.*, 4:48-57. 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*, 4:64-5:7. As well-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.*, ¶¶208-209. 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). *Kennedy's* determination that behavioral data of related files indicates the software is malicious is a determination that the behavioral data indicates that the software is malware according to the '250 Patent. *'250 Patent*, 1:14-17; *Decl.*, ¶¶208-209.

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.*, ¶¶210-215. It also would have been obvious to configure the *Kester* application server module and/or database factory (*see* Claim 1, Element 1.2) to determine whether received data indicates an application is malware based on an identified relationship between application file(s) and different but related files as taught by *Kennedy*. *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 (*see,* Claim 1, Element 1.1) and used by

Kester to determine whether the behavior is indicative of malware as taught by *Kennedy*. *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, 5:13-22. 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.*, ¶211. 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). *Decl.*, ¶¶207-215.

A PHOSITA would have recognized implementation of *Kennedy's* file set tracking and behavior monitoring to be an obvious and predictable improvement to *Kester's* application file monitoring system/method. *Decl.*, ¶¶207-215. 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 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*.; *see also id*., ¶213 (noting a PHOSITA would have recognized identifying when an application file attempts to send another file that it created or modified to another computer system as taught in *Kennedy* is a "hint" in accordance with the teachings of *Kester* that indicates the application should be classified as 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]. *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]. A PHOSITA would have been motivated to apply the teachings of *Kennedy*, which teaches an effective method of tracking and monitoring

different but related files and their behavior, to identify the types of rogue program threats *Kester* was designed to mitigate more effectively. *Decl.*, ¶¶207-215.

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 *Honig* and *Kennedy* renders Claim 3 obvious. *Decl.*, ¶¶207-216.

2. Claim 4

4. A method according to claim 3, wherein if at least one other object to which said object is related is classed as malware, then classifying said object as malware.

Kester teaches that "expected network activity can be determined from network activity by a different but related application" so that "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 classification category (e.g., malware) of a "different but related application," such an implementation would have been obvious. *Decl.*, ¶¶217-221. *Kester* teaches that the "application and any related data" is displayed at the server module and/or database factory and that the related data is used to classify the application. *Kester*, [0134]-[0135], [0120]. A PHOSITA would have understood *Kester's* disclosure of using "any related data" to classify an application encompasses the classification category of a "different but related application." *Kester*, [0134]-[0135], [0120], [0044], [0182]; *Decl.*, ¶¶217-221.

It would have been obvious to configure *Kester's* malware classification process (*see* Claim 1, Element 1.3) based on the teachings of *Kennedy* such that 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*) is classified as malware, the application file is classified as malware. *Decl.*, ¶¶217-221. The combined system/method classifies an application file as malware when at least one different but related application (i.e., another file in the same file set) is classified as malware. *Decl.*, ¶¶217-221. For example, sus-a.exe directly or indirectly creates a new 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.*, ¶¶217-221. The combined method/system classifies sus-b.exe as malware because 1) sus-a.exe created sus-b.exe and 2) sus-a.exe has already been classified as malware. *Decl.*, ¶¶217-221.

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 3. *See* Claim 3; *Decl.*, ¶¶217-221.

Thus, *Kester* in view of *Honig* and *Kennedy* renders Claim 4 obvious. *Decl.*, ¶¶217-222.

3. Claim 5

5. A method according to claim 3, wherein said other objects include the object or similar objects stored on at least some of the remote computers.

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), [0076] (using unexpected behavior to classify/reclassify the application); *see also* Claim 1, Elements 1.1, 1.2, 1.3.

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 3. 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* Claims 3, 4. Based on the *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 on at least some of the remote workstations. *Decl.*, ¶¶223-225.

4. Claim 6

6. A method according to claim 3, wherein said other objects include other objects that are parent objects or child objects or otherwise process-related objects to said object.

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, 4:31-33 ("'touching' can [also] be performed through indirect actions such as launching another process that then touches a file"). Each touched file is added to the set of the file that touched it. *Id.*, 4:33-34. According to the '250 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., '250 Patent*, 11:3-23; *see also id*., 3:31-41, 17:15-67. 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.*, ¶¶226-227.

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 3.

5. Claim 15

15. Apparatus according to claim 13, wherein the base computer is constructed and arranged so that the determining identifies relationships between the object and other objects.

See Claim 3.

6. Claim 16

16. Apparatus according to claim 15, 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 4.

7. Claim 17

17. 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 sonic [sic][6] of said remote computers.

See Claim 5.

8. Claim 18

18. 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 6.

VI. THE *FINTIV* 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.*,

[6] For this proceeding, Petitioner assumes this is a typographical error and should read "some" not "sonic."

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 recently entered the parties' third amended scheduling order, setting **August, 19, 2024** as the trial date for each defendant in five separate lawsuits consolidated only for pretrial issues. Ex. 1021 (Third Amended Scheduling Order), 9. 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 CrowdStrike may occur later. The Board's FWD in this proceeding will be due late June or early July 2024, which is before any trial will proceed against Petitioner or any of the other defendants. Accordingly, even

if the district court jury trials do begin on August 19, 2024 and Petitioner were to try its case before each of the other four defendants, this Board will have issued its FWD before Petitioner's earliest possible jury trial. This factor therefore 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 CrowdStrike served invalidity contentions covering the first six patents (including the '250 Patent). Invalidity contentions for the next six patents are not due until February 21, 2023. Ex. 1021 (Third Amended Scheduling Order), 1. Further, none of the three scheduled *Markman* hearings has been held. *Id*., 3, 5, 6 (setting *Markman* hearings on March 7, 2023, May 3, 2023, and June 2, 2023). And dispositive motions are not due until May 7, 2024. *Id*., 8. This factor weighs in favor of institution.

D. Overlap

Petitioner stipulates that if the Board institutes IPR, Petitioner will not seek resolution in the District Court of any ground of invalidity for the '250 Patent that utilizes *Kester*, *Honig*, or *Kennedy*—the prior art references relied upon in the Proposed Grounds. Petitioner's proposal is comprehensive, narrows the issues, and ensures there is no overlap between the '250 Patent invalidity theories before the

Board and before the District Court. Petitioner's proposed stipulation is far more restrictive than a *Sand Revolution* stipulation, which excludes only the same grounds advanced in the IPR, allowing the Petitioner to rely on the same references in the two tribunals so long as the reliance is not identical. Accordingly, this factor weighs heavily in favor of institution.

E. Same Party

The Parties are the same. However, members of the Board have noted *Fintiv* addresses only the scenario where the petitioner is unrelated to the defendants in litigation, which weighs 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*., IPR2020-00122, 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"). The strength of Petitioner's Proposed Grounds weighs strongly in favor of institution. *Id*. (refusing to deny institution under *Fintiv* based solely on the strength of Petitioner's grounds).

VII. CONCLUSION

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

Respectfully submitted,

ERISE IP, P.A.
BY: */s/ Adam P. Seitz*
Adam P. Seitz, Reg. No. 52,206
Paul R. Hart, Reg. No. 59,646
Hunter A. Horton, *pro hac vice to be submitted*

COUNSEL FOR PETITIONER

VIII. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8

A. Real Parties-in-Interest Under 37 C.F.R. § 42.8(b)(1)

Petitioner and CrowdStrike Holdings, Inc. are the real parties-in-interest.

B. Related Matters Under 37 C.F.R. § 42.8(b)(2)

The '250 Patent is presently the subject of a patent infringement lawsuit filed against Petitioner in the United States District Court for the Western District of Texas Waco Division. *Webroot, Inc. et al. v. CrowdStrike, Inc. et al.*, Case No. 6:22-cv-00241. Petitioner understands the '250 Patent has also been asserted in the following matters:

- *Webroot, Inc. et al. v. Trend Micro Inc.*, Case No. 6:22-cv-00239 (WDTX);
- *Webroot, Inc. et al. v. Sophos Ltd.*, Case No. 6:22-cv-00240 (WDTX); and
- *Webroot, Inc. et al. v. AO Kaspersky Lab*, Case No. 6:22-cv-00243 (WDTX).

Petitioner is also aware of the following matters involving unrelated patents asserted in related district court litigation alongside the '250 Patent:

- *CrowdStrike, Inc. v. Webroot Inc.*, IPR2022-01522 (PTAB Sep. 30, 2022);
- *CrowdStrike, Inc. v. Open Text Inc.*, IPR2023-00126 (PTAB Oct. 31, 2022); and
- *CrowdStrike, Inc. v. Open Text Inc.*, IPR2023-00124 (PTAB Nov. 30, 2022).

C. Lead and Back-Up Counsel Under 37 C.F.R. § 42.8(b)(3)

Petitioner provides the following designation and service information for lead and back-up counsel. 37 C.F.R. § 42.8(b)(3) and (b)(4).

Lead Counsel	Back-Up Counsel
Adam P. Seitz (Reg. No. 52,206) Adam.Seitz@eriseip.com Postal and Hand-Delivery Address: ERISE IP, P.A. 7015 College Blvd., Suite 700 Overland Park, Kansas 66211 Telephone: (913) 777-5600 Fax: (913) 777-5601	Paul R. Hart, (Reg. No. 59,646) Paul.Hart@eriseip.com Postal and Hand-Delivery Address: ERISE IP, P.A. 5299 DTC Blvd., Ste. 1340 Greenwood Village, Colorado 80111 Telephone: (913) 777-5600 Fax: (913) 777-5601 Hunter A. Horton (*pro hac vice to be submitted*) Hunter.horton@eriseip.com Postal and Hand-Delivery Address: ERISE IP, P.A. 7015 College Blvd., Suite 700 Overland Park, Kansas 66211 Telephone: (913) 777-5600 Fax: (913) 777-5601

APPENDIX OF EXHIBITS

Exhibit 1001	U.S. Patent No. 8,418,250 to Morris et al. ("*'250 Patent*")
Exhibit 1002	File History of U.S. Patent No. 8,418,250 ("*'250 File History*")
Exhibit 1003	Declaration of Dr. Wenke Lee ("*Decl.*")
Exhibit 1004	U.S. Patent Application Publication No. 2005/0210035 A1 to Kester et al. ("*Kester*")
Exhibit 1005	U.S. Patent No. 7,225,343 to Honig et al. ("*Honig*")
Exhibit 1006	U.S. Patent No. 7,594,272 to Kennedy et al. ("*Kennedy*")
Exhibit 1007	Plaintiff's Opposition Markman Brief, 22-cv-00243 WDTX, No. 98
Exhibit 1008	Intentionally left blank
Exhibit 1009	Intentionally left blank
Exhibit 1010	Defendants' Opening Markman Brief, 22-cv-00243 WDTX, No. 86
Exhibit 1011	*Intrusion Detection with Unlabeled Data Using Clustering* by Leonid Portnoy, et al. ("*Portnoy*")
Exhibit 1012	U.S. Patent No. 6,944,772 to Dozortsev ("*Dozortsev*")
Exhibit 1013	U.S. Patent No. 6,772,363 to Chess et al. ("*Chess*")
Exhibit 1014	WO 2002/033525 to Shyne-Song Chuang
Exhibit 1015	EP 1,549,012 to Kristof De Spiegeleer
Exhibit 1016	EP 1,280,040 to Alexander James Hinchliffe, et al.
Exhibit 1017	U.S. Patent Publication No. 2004/0153644 to McCorkendale ("*McCorkendale*")
Exhibit 1018	U.S. Patent No. 7,516,476 to Kraemer ("*Kraemer*") U.S. Patent No. 7,516476 to Kraemer ("*Kraemer*")
Exhibit 1019	Harold S. Javitz et al., The NIDES Statistical Component: Description and Justification" ("*The NIDES Statistical Component: Description and Justification*")
Exhibit 1020	U.S. Patent No. 7,448,084 to Apap et al.
Exhibit 1021	Order Granting Third Amended Scheduling Order, 22-cv-00243 WDTX, No. 142 ("*Third Amended Scheduling Order*")

CERTIFICATION OF WORD COUNT

The undersigned certifies pursuant to 37 C.F.R. §42.24 that the foregoing Petition for *Inter Partes* Review, excluding any table of contents, mandatory notices under 37 C.F.R. §42.8, certificates of service or word count, or appendix of exhibits, contains 13,626 words according to the word-processing program used to prepare this document (Microsoft Word).

Dated: December 29, 2022

Respectfully submitted,

BY: */s/ Adam P. Seitz*

Adam P. Seitz, Reg. No. 52,206
Adam.Seitz@eriseip.com
Erise IP, P.A.
7015 College Blvd., Suite 700
Overland Park, KS 66211
P: (913) 777-5600
F: (913) 777-5601

COUNSEL FOR PETITIONER

CERTIFICATE OF SERVICE ON PATENT OWNER
UNDER 37 C.F.R. § 42.105(a)

Pursuant to 37 C.F.R. §§ 42.6(e) and 42.105(b), the undersigned certifies that on December 29, 2022, a complete and entire copy of this Petition for *Inter Partes* Review and Exhibits were provided via Priority Mail Express to the Patent Owner by serving the correspondence address of record for the '250 patent:

Merchant & Gould PC
P.O. Box 2903
Minneapolis, MN 55402-0903

Respectfully submitted,

ERISE IP, P.A.

BY: */s/ Adam P. Seitz*
Adam P. Seitz, Reg. No. 52,206
Adam.Seitz@eriseip.com
Erise IP, P.A.
7015 College Blvd., Suite 700
Overland Park, KS 66211
P: (913) 777-5600
F: (913) 777-5601

COUNSEL FOR PETITIONER