UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.
Petitioner

v.

WEBROOT INC.
Patent Owner

Case No. IPR2023-01053
Patent No. 11,409,869

PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 11,409,869

TABLE OF CONTENTS

I. INTRODUCTION 1

II. SUMMARY OF THE '869 PATENT 1

A. DESCRIPTION OF THE '869 PATENT 1
B. SUMMARY OF THE PROSECUTION HISTORY 4
C. CLAIM CONSTRUCTION UNDER 37 C.F.R. § 42.104(B)(3) 7
D. LEVEL OF SKILL OF A PERSON HAVING ORDINARY SKILL IN THE ART 8

III. REQUIREMENTS UNDER 37 C.F.R. § 42.104 8

A. GROUNDS FOR STANDING UNDER 37 C.F.R. § 42.104(A) 8
B. IDENTIFICATION OF CHALLENGE UNDER 37 C.F.R. § 42.104(B) 9

IV. THERE IS A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE 9

A. GROUND 1: *KUMAR*, OR ALTERNATIVELY, *KUMAR* IN VIEW OF *TAHAN* RENDERS CLAIMS 1-59 OBVIOUS 9

V. DISCRECTIONARY DENIAL IS NOT WARRANTED 64

A. THE *GENERAL PLASTIC* FACTORS FAVOR INSTITUTION 64
B. THE *ADVANCED BIONICS* FRAMEWORK FAVORS INSTITUTION 68
C. THE *FINTIV* FACTORS FAVOR INSTITUTION 70

VI. CONCLUSION 75

VII. MANDATORY NOTICES UNDER 37 C.F.R. § 42.8 77

A. REAL PARTIES-IN-INTEREST UNDER 37 C.F.R. § 42.8(B)(1) 77
B. RELATED MATTERS UNDER 37 C.F.R. § 42.8(B)(2) 77
C. LEAD AND BACK-UP COUNSEL UNDER 37 C.F.R. § 42.8(B)(3) 77

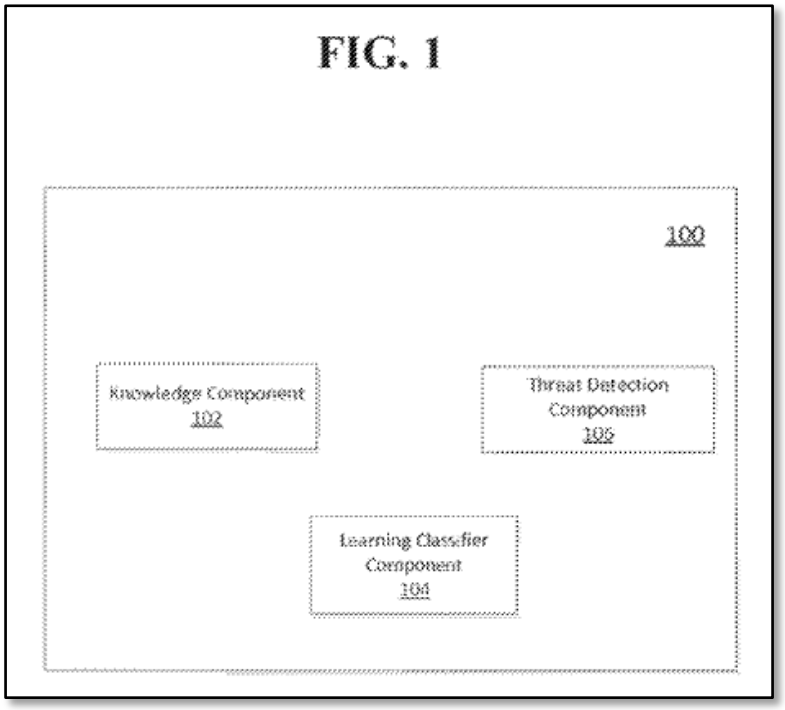
I. INTRODUCTION

CrowdStrike, Inc. ("Petitioner") respectfully requests *inter partes* review ("IPR") of claims 1-59 (collectively, the "Challenged Claims") of U.S. Patent No. 11,409,869 (the "'869 Patent").

II. SUMMARY OF THE '869 PATENT

A. Description of the '869 Patent

Generally directed "to threat detection of executable files[,]" the '869 Patent purports to have improved on prior art threat detection analytics which focused on file behavior to identify malicious code blocks. '869 Patent (Ex. 1001), 1:31-45, 2:23-36. In contrast to behavior analysis, the '869 Patent analyzes threats by "evaluating static data points extracted from an executable file" (e.g., file size, file name, file version, etc.) and uses a "trained classification system" to analyze this data. *Id*., 2:36-53, Table 1.1. An exemplary system which implements the disclosed threat detection techniques is depicted below:



Id., Fig. 1.

In embodiments, this system analyzes a portable executable (PE) file for threats. *Id*., 3:46-54 (noting the system is not limited to PE files and can analyze "any file or program that executes a task according to an encoded instruction"), 7:5-16. To analyze threats, knowledge component 102 "collect[s] data for use in building, training and/or re-training a learning classifier to evaluate executable files." *Id*., 3:54-60, 4:29-44. This includes collecting any number of static data points, such as:

> header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples.

Id., 5:62-6:44, Table 1.1.

Learning classifier component 104, which is trained with known threat, benign, and unwanted data, then evaluates the executable file "using the information provided by the knowledge component 102" through "programming operations or machine learning processing." *Id*., 7:26-34, 8:4-7 (listing examples of programming and machine learning processes). As part of this evaluation, the learning classifier can "adaptively set features…based on the static data points" and "controllably select (e.g., turn on/off) features for evaluation." *Id*., 7:43-48. "In one instance where a static data point extracted for evaluation is a file size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether [] the file size of the file is within a certain range." *Id*., 7:48-52.

To make the static data points suitable for the above processing operations, learning classifier 104 processes this data. *Id*., 8:8-10. For example, "string sequence data as well as byte sequence data may be parsed and processed as n-grams" such that "the unique n-grams string and/or byte sequences are then used as input to a generated feature vector." *Id*., 8:16-23. In the Challenged Claims, this feature vector is "input to a support vector machine (SVM)" or a "machine learning model" for categorization. *Id*., 9:27-35. Finally, threat detection component 106 evaluates the results of the learning classifier and makes a "determination as to whether the executable file is malicious, benign or a potentially unwanted file/application." *Id*., 9:55-59, Fig. 3.

B. Summary of the Prosecution History

Application No. 16/791,649 (the "'649 Appl.") which issued as the '869 Patent was filed on February 14, 2020 and claims priority to a continuation application filed on May 12, 2015, which issued as the parent '844 Patent. Petitioner applies May 12, 2015, as the priority date for the Challenged Claims, which means the first inventor to file provisions of the Leahy-Smith America Invents Act (AIA) apply.

1. '844 Parent Prosecution

The claims of the '844 Patent were rejected several times over the prior art, including several rejections involving US 8,709,924 to Wojnowicz et al. ("Wojnowicz"). '844 FH (Ex. 1006), 92-108, 147-166, 209-229, 265-284. Per Applicant, Wojnowicz:

> relates to identifying malware using wavelet decomposition of software entropy. Specifically, Wojnowicz teaches receiving a set of data files. Each of the data files is split into non-overlapping file chunks of fixed length. The non-overlapping file chunks are represented as an entropy time series that reflects an amount of entropy across the file chunks for each file. A wavelet transform is applied to the corresponding entropy time series to generate an energy spectrum characterizing, for the file, an amount of entropic energy at multiple scales of code resolution. A determination is then made as to whether the file is likely to be malicious based on the energy spectrum. Wojnowicz, col. 1, 11. 23-31.

Id., 134. Applicant first amended its claims to "clarify that the static data points extracted from the received executable file correspond to points of reference (in the file content) that are used to evaluate the file" such that Wojnowicz's file chunks

split into "a fixed length boundary (e.g., 256 bytes)" "do not correspond to particular sections or portions of the file." *Id*., 135. The Examiner rejected Applicant's arguments and issued a final rejection, explaining an example static data point is "string information" and that "Wojnowicz computes entropy of a string of hexadecimal characters." *Id*., 147-149 (further noting "the location of the string within the executable file is static").

Applicant filed an RCE and further amended the claims to state that "the one or more points of reference are defined by the executable file." *Id*., 185-186. Applicant argued that Wojnowicz's fixed length boundaries for its chunks "occur randomly" and cannot be defined by the executable file. *Id*. The Examiner again rejected these arguments, noting "Wojnowicz splits the file into sections 256 byte apart, thus a point of reference is defined at each portion of the file located at each consecutive 256 byte location. Expressed differently, this point of reference is defined by the executable file because it is based upon the executable file and the number of sequential bytes counted in the executable file to define the each point." *Id*., 211. Applicant thereafter made a similar amendment, requiring "static data points are points of reference *representing sections of the executable file*,[1] " suggesting because Wojnowicz's chunks "may comprise data from different sections

[1] All emphasis added unless noted otherwise.

of the file," these chunks do not represent sections of the file, which was similarly rejected by the Examiner. *Id*., 249, 265-268.

Applicant filed another RCE, removing the "points of reference" language and adding "*the plurality of static data points represent predefined character strings in the executable file*." *Id*., 305. Applicant argued that Wojnowicz's "fixed-length file chunks do not represent predefined character strings" where "each fixed-length file chunk is, itself, a random (e.g. undefined) character string having no initial significance to the system of Wojnowicz." *Id*., 312.

The Examiner rejected the amended claims with new references, and Applicant presented further amendments, adding the "generating a feature vector…" limitation found in the issued claims including the language requiring a feature of the feature vector to be "selectively turned on or off based on whether a value of one or more static data points…is within a predetermined range." *Id*., 372. This positioned the claims for allowance. *Id*., 387-395.

2. '869 Prosecution

The independent claims of the '869 Patent omit several of the concepts recited in the independent claims of the parent '844 Patent, such as requiring "static data points" to be "predefined character strings" or limiting extraction of static data points to be without unpacking or decrypting. In contrast, the challenged independent claims were amended to include the following limitation (also found in the parent

'844 Patent claims, as noted above) to traverse the cited art of record: "wherein one or more features of the feature vector are selectively turned on or off *based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range*." '869 FH (Ex. 1002), 217, 226. This limitation purportedly differentiated the claimed invention from both "prior art discussing simply disregarding/ignoring features of a data set as a means to avoid skewing classification" (*see id.*, 121 discussing US2011/0172504 to Wegerich) and "selectively turning on/off one or more features…based on the number of classifiers associated with the features for the benefit of efficiency" (*see id.*, 259 discussing US2010/0082642 to Foreman).

A Notice of Allowance issued thereafter, citing the above limitation as the reason for allowance. *Id*., 253-259.

C. 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. 304, *Response Markman Brief*), 56-60. For the purposes of this proceeding only, Petitioner proposes the Board adopt PO's proposal. *See Western Digital Corporation v. Spex Technologies, Inc.*, IPR2018-00084, Paper

14, *11 (PTAB Apr. 25, 2018) (finding Petition based on claim constructions urged by Patent Owner satisfies the claim construction requirements and Petitioner is not required to express its subjective agreement regarding correctness of the proffered claim construction or take ownership of the construction); *see also General Electric v. Vestas,* IPR2018-00928, Paper 9, *16-17 (PTAB Nov. 5, 2018).[2]

D. Level of Skill of a Person Having Ordinary Skill in the Art

As of May 12, 2015, 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. Dr. Jakobsson Declaration ("*Dec*.") (Ex. 1003), ¶¶50-52.

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

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

Petitioner certifies the '869 Patent is eligible for IPR.

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

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 Grounds of Unpatentability	Exhibits
Ground 1: Claims 1-59 are **obvious** under § 103(a) over U.S. Patent Publication No. 2012/0317644 to Kumar et al. ("*Kumar*") or, Alternatively, Claim 1-59 are **obvious** under § 103(a) over *Kumar* in view of Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features by Tahan et al. ("*Tahan*").	Ex. 1004; Ex. 1005

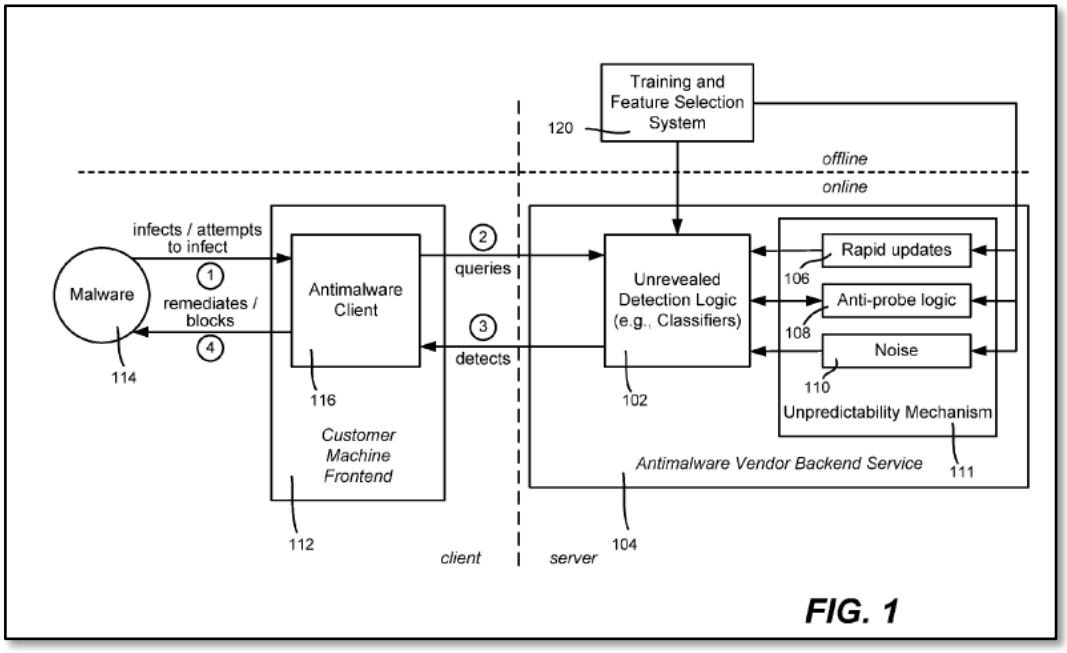
IV. THERE IS A REASONABLE LIKELIHOOD THAT THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. Ground 1: *Kumar*, or Alternatively, *Kumar* in view of *Tahan* renders Claims 1-59 Obvious

Overview of Kumar

Kumar published on December 13, 2012 and therefore qualifies as prior art to the '869 Patent under 35 U.S.C. §§ 102(a)(1) and 102(a)(2). *Kumar* (Ex. 1004). *Kumar* was not cited or considered during prosecution of the '869 Patent.

Kumar discloses a threat detection system which utilizes "antimalware detection logic [] maintained and operated at a backend service…for purposes of malware detection." *Kumar*, Abstract, [0005], [0018]. This system is depicted below:



Id., Fig. 1.

In operation, anti-malware client component 116 on the frontend machine 112 receives an unknown file and performs processing before the file is allowed to run. *Id*., [0022]-[0023], Fig. 5. In embodiments, component 116 extracts and sends information comprising characteristics and features of potential malware executables to the backend service in the form of a "telemetry report," optionally including the following static data points:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Id., [0026]-[0028]; *see also id*., [0035] (noting "entropy" "also may be used as a feature").

The backend service 104 receives this data and processes it using detection logic 102. *Id*., [0026]. Detection logic 102 "includes one or more trained telemetry report classifiers that determine, in part, whether the features and characteristics are indicative of malware." *Id*.; *see also id*., [0027] (listing exemplary classifiers, including a "support vector machine"). To accomplish this, feature data is preprocessed into a feature vector, the classifier analyzes this feature vector based on training data (*see* Fig. 2 for training process), and outputs a result, which may be "a probability" that the file "belong to a particular malware family, the generic malware family, a particular benign family, or the generic benign family." *Id*., [0036]-[0045], [0052], [0068]-[0071], Fig. 5.

Because *Kumar*, like the '869 Patent, discloses systems and methods for threat detection of executable files, *Kumar* is in the same field of endeavor as the '869 Patent. *Compare Kumar*, Abstract, [0005], [0018], [0036]-[0045], [0052], [0068]-[0071], Fig. 5 *with* '869 Patent, 1:31-45, 2:23-53, 3:46-60, 4:29-44, 7:26-34, 8:4-7; *Dec*., ¶¶55-60. *Kumar* is also directed at solving similar problems, such as detecting threats without decrypting and unpacking unknown files and/or using static data points to analyze a file. *Compare Kumar*, [0039], [0054] (discussing analysis without unpacking), [0028] (identifying exemplary static data points for analysis), *with* '869 Patent, 1:31-36, 2:23-53, 12:11-45, claims 2, 11, 16, 19, 33, 47; *Dec*., ¶¶55-60.

Overview of Tahan

Tahan (Ex. 1005) was publicly available by at least April 2012 and therefore qualifies as prior art to the '869 Patent under 35 U.S.C. §§ 102(a)(1) and 102(a)(2). *Munford Dec.* (Ex. 1008), ¶¶6-10; *Dr. Yee Dec.* (Ex. 1015), ¶¶28-39. *Tahan* was not cited or considered during prosecution of the '869 Patent.

Tahan discloses "methods, based on machine learning, to detect malware in executable files without any need for preprocessing, such as unpacking or disassembling." *Tahan*, 949. In embodiments, *Tahan's* methods employ "Mal-ID[,]" a "static (form-based) analysis methodology that uses common segment analysis…to detect malware files." *Id*. Mal-ID's methodology is separated into two

phases: (1) a "Setup Phase" and (2) a "Malware Detection Phase." *Id.*, 955-956.

These phases are shown generally below:

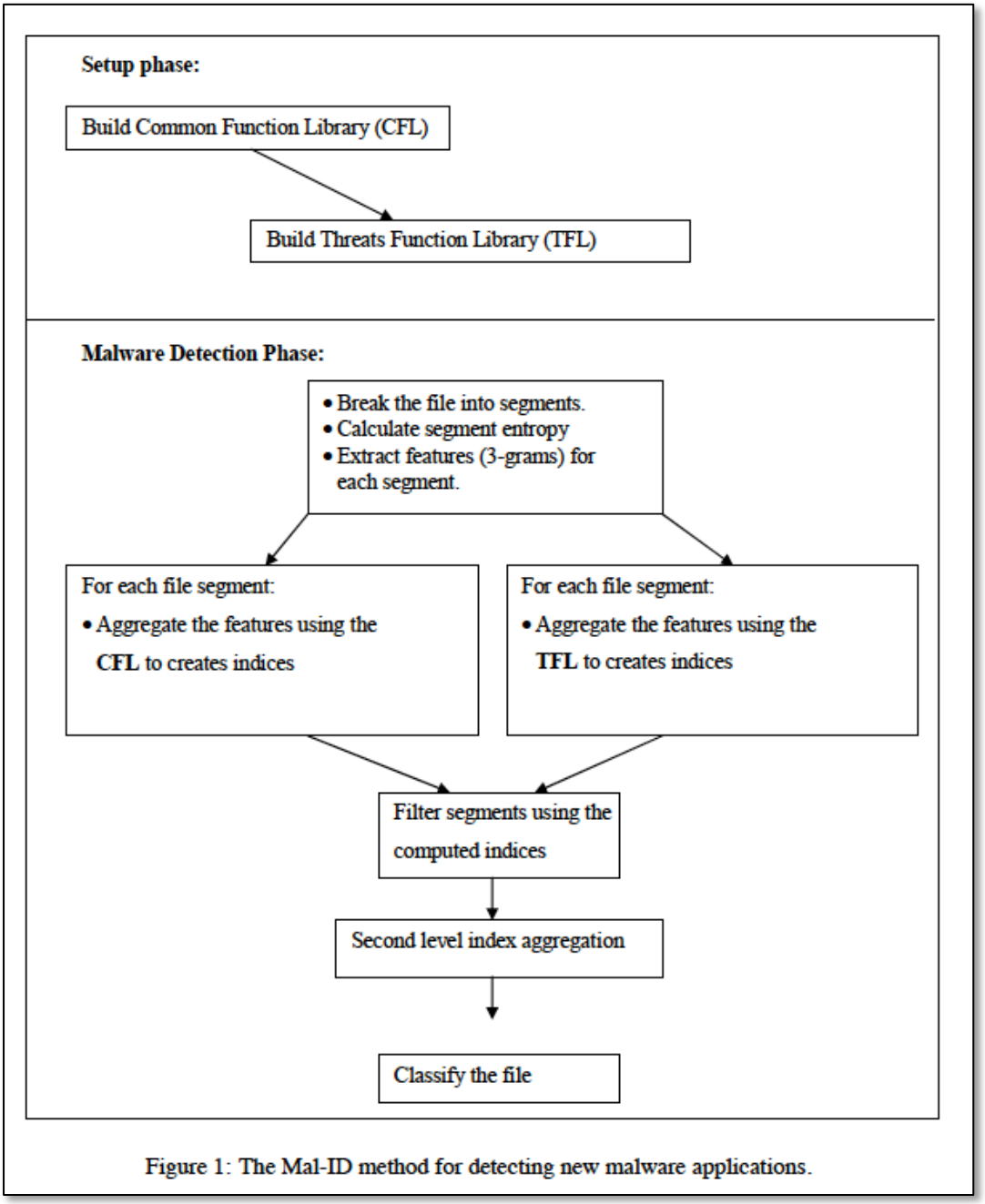


Figure 1: The Mal-ID method for detecting new malware applications.

Id., 956.

During Setup, *Tahan* builds the repositories requisite for the Detection phase, including a "Common Function Library" (CFL) which contains benign files and a "Threat Function Library" (TFL) which contains malware files. *Id*., 955. Then, during Detection, a file is broken into 64-byte segments, these segments are broken into 3grams (three consecutive bytes), and features are derived for each segment including "Spread, MFG, and Entropy." *Id*., 955-959. These features are "meta-features" "used to represent features of features (features of the basic 3-grams)." *Id*., 957. The system then uses a detection algorithm to a classify a file as benign or malware. *Id*., 958-958; *see also id*., 960 (providing pseudocode for this algorithm). During evaluation, this algorithm removes irrelevant segments by setting thresholds for segment entropy values, including EntropyLow and EntropyHigh such that entropy's falling below the low threshold or above the high threshold are discarded. *Id*., 958, 960.

Because *Tahan*, like the '869 Patent, discloses systems and methods for threat detection of executable files, *Tahan* is in the same field of endeavor as the '869 Patent. *Compare Tahan*, 949, 955-959, Figures 1 and 2 *with* '869 Patent, 1:31-45, 2:23-53, 3:46-60, 4:29-44, 7:26-34, 8:4-7; *Dec*., ¶¶61-65. *Tahan* is also directed at solving similar problems, such as detecting threats without decrypting and unpacking unknown files. *Compare Tahan*, 949, 975 (discussing analysis without

unpacking), [0028] (identifying exemplary static data points for analysis), *with* '869 Patent, 1:31-36, 2:23-53, 12:11-45, claims 2, 11, 16, 19, 33, 47; *Dec*., ¶¶61-65.

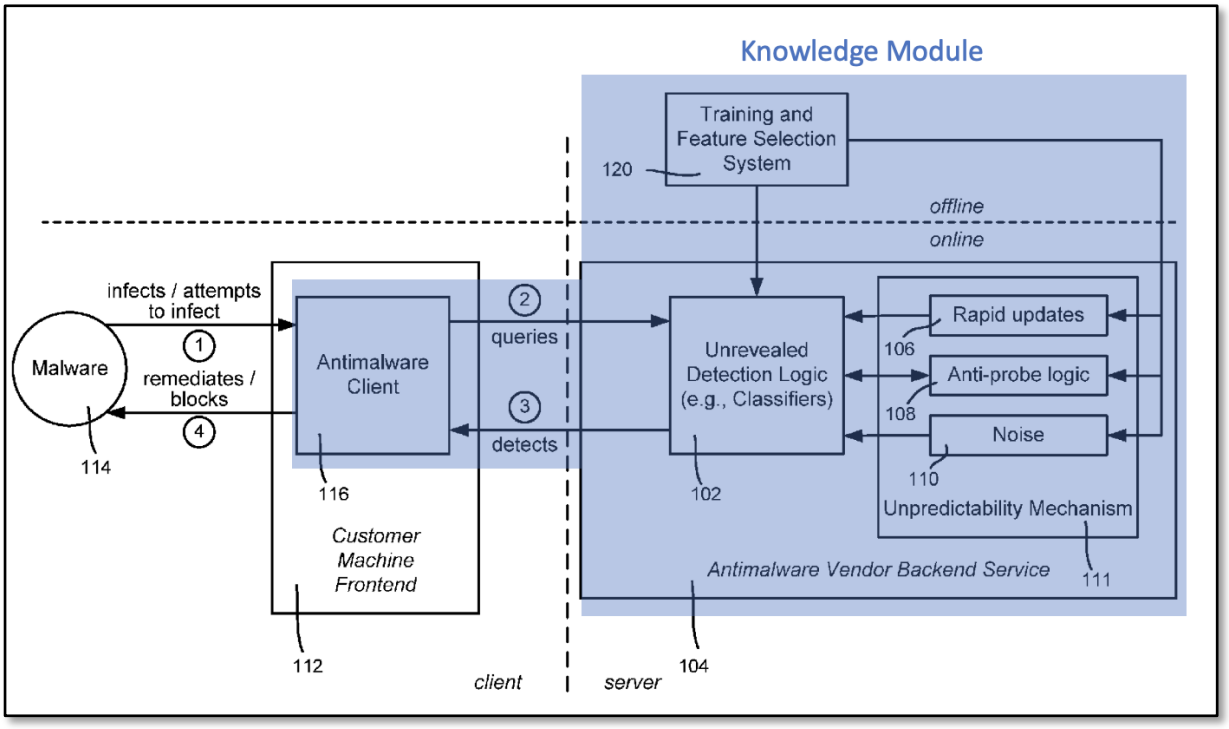
1. Claim 1

1[Pre]. A computer-implemented method comprising:

Kumar discloses a computer-implemented method where "processing determines whether the data associated with the malware-related query corresponds to detected malware[.]" *Kumar*, [0005]; *see also id*., Claim 1 ("**In a computing environment, a method performed** at least in part on at least one processor...").

[1(a)] identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;

In *Kumar's* methods, *Kumar* discloses an antimalware detection system (i.e., *knowledge module*) that *identifies static data points that may be indicative of either a harmful or benign executable file*. As shown below, *Kumar's* knowledge module includes an antimalware client 116 running on a customer's local machine and an antimalware backend service 104 running on a server:



Id., Fig. 1 (annotated). While the '869 Patent does not define a "knowledge module," it generally explains that a "knowledge component 102 is one or more storages, memories, and/or modules that continuously collects and manages data that may be used to detect threats in files such as executable files." '869 Patent, 1:59-61. The '869 Patent's components "may be spread across multiple devices." *Id.*, 3:4-24, 3:34-42 (noting the scale of the system can vary and include additional or fewer components than those described in Figure 1, including a client and server architecture with processing spread across that architecture). A PHOSITA would have understood that *Kumar's* antimalware detection system therefore constitutes a "knowledge module" as claimed. *Dec.*, ¶¶67-76.

During a training process, *Kumar's* knowledge module identifies several sets of static data points to train a classifier used to predict whether an unknown file is malware or benign. For example, in one implementation, "[r]aw telemetry logs 222 (reports) and analyst labels 224 are input to [training and feature selection] system 120." *Kumar*, [0051]. "Some example features derived from the telemetry data and used to train the classifiers" are identified in the table below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Id., [0028] (further noting an additional "Boolean feature" can be used "to indicate if a particular attribute is blank or null"). Additionally, "[s]ome measure of the differences in the file names such as the entropy of the observed filenames also may be used as a feature." *Id*., [0035]. "The features described in the feature table can be used to classify if an individual report corresponds to a malicious file." *Id*.

In another implementation, *Kumar's* system "uses system state variables, unpacked strings, API system call trigrams, and API system calls plus an individual parameter value as potential features[.]" *Id*., [0055]. *Kumar* notes "that the features described herein are only examples, and the classifier is not limited to these types of features, as others (e.g., file geometry, file reputation and so forth) may be used instead of or in addition to some or all of the ones exemplified herein." *Id*.

In each implementation, training and feature selection system 120 "includes various processing blocks including a feature selection component 226" that "determines a subset of the total features to be used for classification." *Id*., [0051], [0055]. Here, for example, "a feature selection algorithm attempts to choose the 3,000 'best' features per family, and 12,000 'best' features for the generic malware and benign families (where 'best' may be determined according to an appropriate ranking metric)." *Id.* Thus, *Kumar's* selected features may be indicative of either a malicious (i.e., *harmful*) or benign executable file. Indeed, once these features are identified and collected, a "labeled dataset is constructed from the analysts' labels 224" and the feature data and is "used to train a classifier model 232[.]" *Id*., [0052] (describing using telemetry feature data for training); [0055] (describing using file execution feature data for training). To train the model, a preprocessing step converts the raw data to the feature vector used to train the classifier. *Id*., [0052], [0055]. The output of the training system comprises a classifier model "(e.g., a set of weights or

parameters), which can be used to **predict if an unknown file belongs to a particular malware family, a generic malware class, a particular benign family, or a generic benign class**." *Id*.; *see also id*., [0068], Fig. 5 (steps 503 and 505).

Many of *Kumar's* features, including file names, file description, file type, version information, certificate information, boolean values, entropy, etc. are identified as *static data points* by the '869 Patent. The '869 Patent discloses that "static data points include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples." '869 Patent, 5:61-67. Moreover, the '869 Patent provides Table 1.1 which "identifies some of the static data points identified for analysis of an executable file[.]" *Id.*, 6:15-18. Table 1.1 is reproduced below:

TABLE 1.1

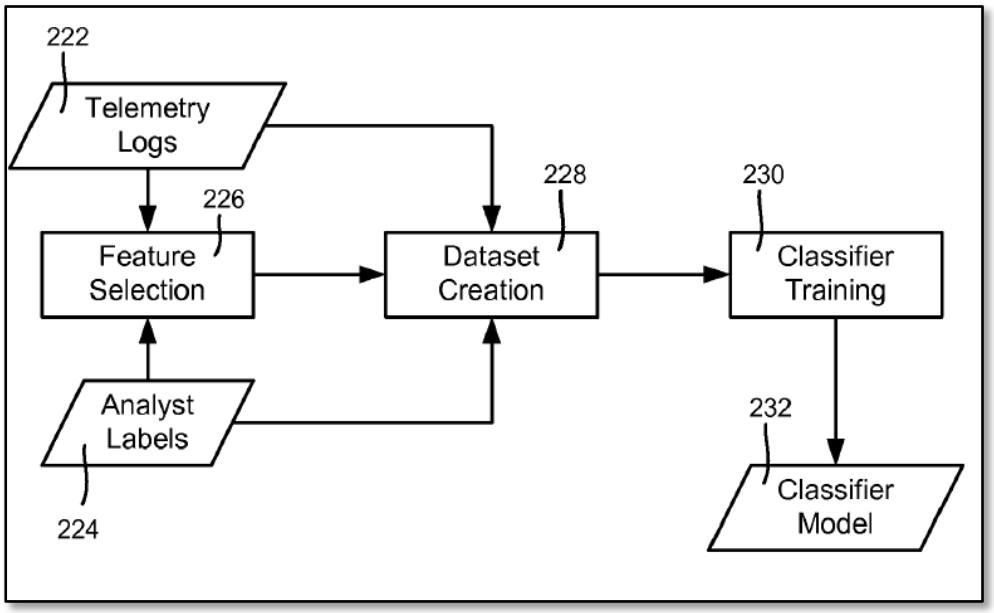
Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heapr	file OS	special build	Code Characteristics Anomaly
size of stackr	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

Id., Table 1.1. Therefore, *Kumar's* identified features include *static data points* within the meaning of the '869 Patent. *Dec.*, ¶¶44-47, 76 (noting *Kumar's* identified features also fall within the description of static data points during the '844 parent patent's prosecution—"static data points extracted from the received executable file correspond to points of reference (in the file content) that are used to evaluate the file").

[1(b)] associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

During *Kumar's* training process, several features (i.e., static data points) are identified and collected, and training system 120 constructs a labeled dataset from these features and the analyst labels to train the classifier. *Supra*, Claim 1(a)

(identifying training system 120 as part of the "knowledge module"); *Kumar*, [0052], [0055]. An example of this process is shown below in Figure 2:



Id., Fig. 2, Fig. 3 (identifying same process for file execution feature data). In this process, *Kumar* associates the identified static data points with categories of files, such as harmful and benign files.

For example, as depicted in Figure 2 above, "[r]aw telemetry logs 222 (reports) and analyst labels 224 are input to the system 120, which includes various processing blocks including a feature selection component 226, a dataset creation component 228, and a classifier training component 230." *Id*., [0051]. During feature selection, "the feature selection component 226 determines a subset of the total features to be used for classification." *Id.*, [0051]. "In one example implementation, a feature selection algorithm attempts to choose the 3,000 'best' features per family,

and 12,000 'best' features for the generic malware and benign families (where 'best' may be determined according to an appropriate ranking metric)." *Id.*, [0055], [0063] (noting the feature selection process is the same for the telemetry and file execution implementations).

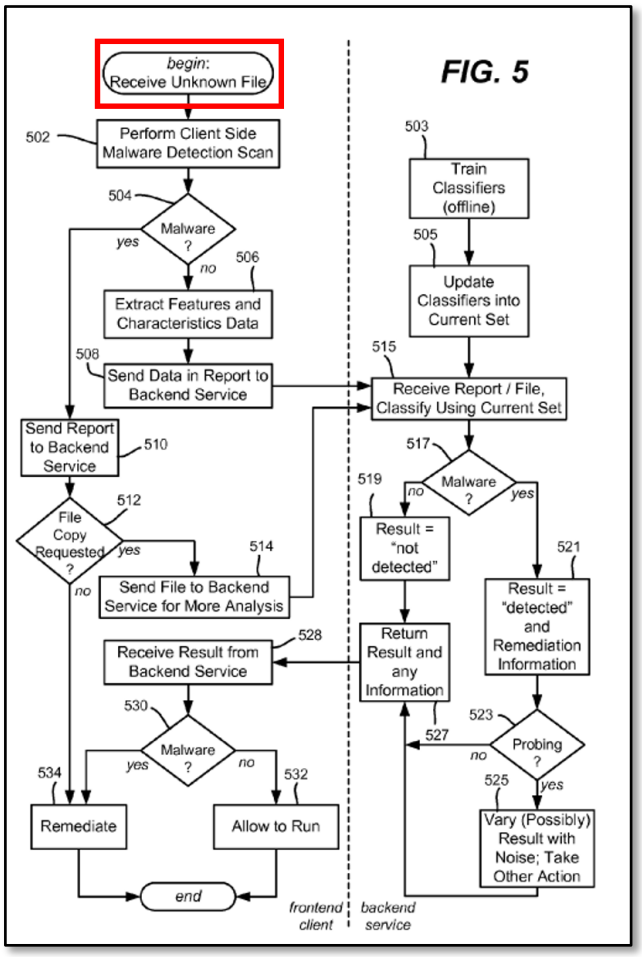
"Thus, a ranked list of potential features for each family is computed, and used to choose the top K (3000, 12000) features for the family." *Id*., [0056]. "**These top K features are a mixture of different types of features, depending on how well they discriminate the one family from other malware and benign file classes** (e.g., other malware families, generic malware family, other generic families, and generic benign family)." *Id.* "The output of the training system comprises a classifier model 232 (e.g., a set of weights or parameters), which can be used to predict if an unknown report was generated by a particular malware family, a generic malware class, a particular benign family, or a generic benign class." *Id.*, [0052].

Therefore, *Kumar's* knowledge module *associates the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files*. *Dec.*, ¶¶77-80.

[1(c)] identifying an executable file to be evaluated;

Kumar discloses that "[w]hen potential malware is received at the customer machine front-end 112, the potential malware is processed before being allowed to run, so as to detect whether the potential malware is actual malware 114." *Kumar*,

[0022]. As shown in Figure 5, the frontend client identifies an unknown executable file for evaluation after it is received at the front-end machine:



Id. at Fig. 5 (emphasis added); *see also id.*, [0026] (describing sending features for "executables" to a backend service), [0029] (backend service receiving a "report of the executable"), [0032] (transmitting telemetry reports for received "[e]xecutable files" to backend service).

[1(d)] extracting, by the knowledge module, a plurality of static data points from the identified executable file;

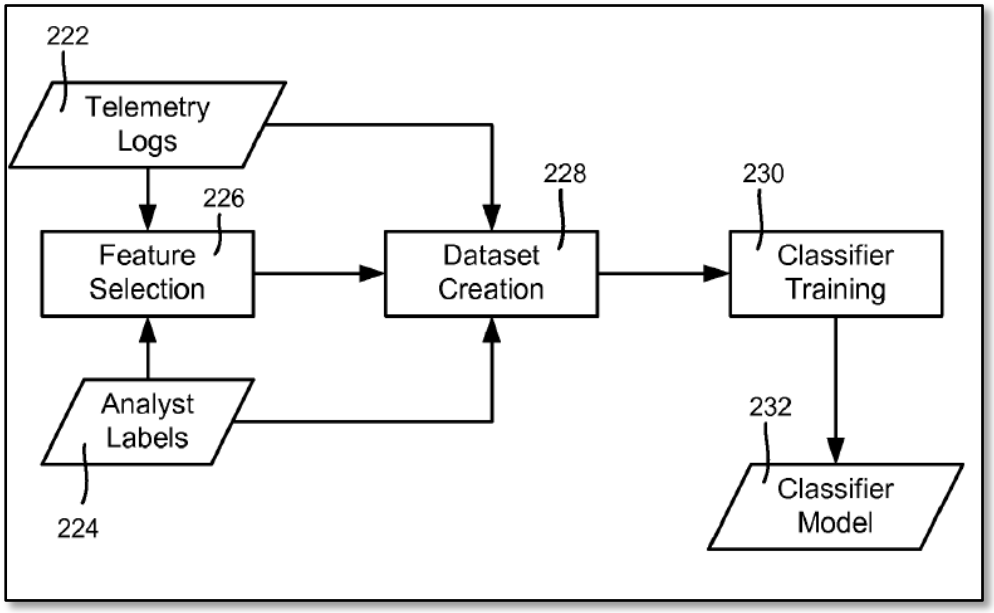
As discussed above, *Kumar's* knowledge module includes a frontend client 116. *Supra,* Claim 1(a). After a potential malware file is identified, this frontend client "may **extract**" feature data to send to a backend service. *Kumar*, [0006]; *see also id.*, [0026] ("features and characteristics may be **extracted from the file** data by the antimalware client component 116, sent to the backend service 104, (e.g., in an xml report), and processed at the backend malware detection logic 102"), [0067] ("**extract** the features and any other characteristics data from the file"), Fig. 5 (step 506), claim 4. As described above in Claim 1(a), this feature data includes static data points, such as file names, file description, file type, version information, certificate information, boolean values, entropy, etc. identified in the table below:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Supra, Claim 1(a) (explaining this feature data falls within both the specification and parent '844 patent file history disclosures of "static data points").

[1(e)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

As described above, *Kumar* discloses a training process where a classifier is trained with training data, such as prior telemetry logs and analyst labels. *Supra*, Claim 1(a)-1(b). Specifically, "a labeled dataset is constructed from the analysts' labels 224 and the telemetry logs 222, and used to train a classifier model 232 using various algorithms." *Kumar*, [0052]. This process is identified below in Figure 2:



Id., Fig. 2.

Kumar's training data comprises files known to fit into different families of malware and benign files (i.e., *files known to fit in one of the plurality of categories*

of files). Namely, *Kumar* discloses its malware classifiers can "be trained" with "labeled files." *Id*., [0047]. In an example, *Kumar* discloses classifying Rbot malware by training Rbot binary classifiers with features from "fixed or randomly selected subset of 50,000 labeled Rbot files." *Id*. To train the classifier with these files, as explained above, "a feature selection algorithm attempts to choose the 3,000 'best' features per family, and 12,000 'best' features for the generic malware and benign families (where 'best' may be determined according to an appropriate ranking metric)." *Id.*, [0055]. In this way, "a ranked list of potential features for each family is computed, and used to choose the top K (3000, 12000) features for the family" such that these features "discriminate the one family from other malware and benign file classes (e.g., other malware families, generic malware family, other generic families, and generic benign family)." *Id.*, [0056]. "The output of the training system comprises a classifier model 232 (e.g., a set of weights or parameters), which can be used to predict if an unknown report was generated by a particular malware family, a generic malware class, a particular benign family, or a generic benign class." *Id.*, [0052].

Kumar's training process is consistent with the embodiments disclosed in the '869 Patent. For example, the '869 Patent explains that "[t]o classify executable files…knowledge component 102 collects static data on a large variety of executable files (e.g., PE files)." '869 Patent, 4:29-31. This static data "yields a knowledgebase

that may be used to periodically train a classifier." *Id*., 4:39-43; *see also id*., 7:34-42 (describing "the information used to train/re-train the learning classifier" including "static data on the robust collection of executable files"), 11:60-67 (describing method 300, including operation 302 which "builds the knowledge base from the data collected and evaluated related to known" files). Thus, the classifier is trained with *training data comprising files known to fit into one of the plurality of categories of files*. *Dec.*, ¶¶84-87.

In a malware detection phase, the features extracted from the unknown file (i.e., the plurality of static data points identified in Claim 1(a) and 1(d)) are transmitted to the backend as a telemetry report and processed by detection logic 102:

> [I]n this example step 506 is performed at the client to extract the features and any other characteristics data from the file, which is placed into a report. Step 508 sends the report in a query to the backend service…Step 515 represents receiving the report (or file if sent via step 514) from the frontend client at the backend service, and classifying the report as indicative of malware or not based upon the data (of file content) therein.

Id., [0067]; *see also id.*, [0026] (noting detection logic 102 includes "trained telemetry report classifiers"), Fig. 5.

A PHOSITA would have understood or found obvious that the extracted telemetry data (i.e., *plurality of static data points*) is converted into feature vectors for processing by "one or more trained telemetry report classifiers that determine, in

part, whether the features and characteristics are indicative of malware[.]" *Dec*., ¶¶88-99; *Kumar*, [0026]. For example, *Kumar* discloses exemplary telemetry classifiers, several of which require feature vectors as input, such as a "support vector machine" (SVM). *Id*.; *Kumar*, [0027] (disclosing exemplary classifiers including "logistic regression, naive Bayes, support vector machine, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like"), [0039]. A PHOSITA would have been motivated and would have had success using classifiers which utilize feature vectors as input, such as an SVM, as disclosed in *Kumar* because these are well-known classifiers for threat detection with widely appreciated benefits. *Dec*., ¶¶88-99 (describing SVM benefits including "good generalization," solution to the high dimensionality concerns, and overall better performance than other traditional AI techniques).

Indeed, *Kumar* already discloses generating feature vectors from static data points to train a classifier, such that a PHOSITA would have understood that during detection, feature vectors would similarly be generated in the same way to facilitate classification. *Dec*., ¶¶88-99; *Kumar*, [0052] ("[A] preprocessing step (not shown) is usually needed to convert the raw telemetry XML data to the actual **feature vector**"), [0055] ("convert the raw data recovered by the system to the actual **feature vector**").

[1(e)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

As described above, *Kumar* discloses or renders obvious generating a feature vector from the extracted features during a malware detection phase. *Supra*, Claim 1(e). Exemplary extracted features are reproduced below in the following table:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Kumar, [0028]. *Kumar* additionally discloses a process for turning off features during evaluation. Namely, *Kumar* discloses that "**one or more other features** may be used to indicate if a particular attribute is blank or null." *Id*. For example, in the table above, if the "organization" feature "is null, a boolean feature is set to true." *Id*. Here, a PHOSITA would have understood that a separate boolean feature associated with the "organization" attribute set to "true" indicates that the malware

file did not contain "organization" data representing the "[m]anufacturer of the binary file." *Dec*., ¶¶100-102. By setting the boolean feature to "true," *Kumar's* system discards (i.e., *turns off*) the "organization" feature such that only the Boolean feature is input to the classifier to indicate the absence of organization data. *Id*. Indeed, a PHOSITA would have understood this boolean feature would be used in lieu of the "blank or null" organization feature because it would be redundant to introduce both a blank/null feature state ***and*** a boolean feature indicating this feature was blank/null for classification. *Id*.

In another example, if *Kumar's* system extracted the above static telemetry data and the "Signature Type" was blank/null—meaning the file was not signed—then a boolean value for the "Signature Type" feature would similarly be set to true, turning off the "Signature Type" feature and introducing only the boolean feature for classification. *Dec*., ¶103. In this example, because there was no "Signature Type" data in the file being analyzed, other associated features would similarly have boolean values set to true and would therefore be turned off. *Id*. Specifically, the "Signer Name" and "Signing Authority" features would have Boolean values set to true, thereby turning these features off given that the file was not signed. *Id*. (noting an unsigned file has no data concerning what organization signed the file (Signer Name), nor what certificate authority issued the signature (Signing Authority)).

a. "within a predetermined range" (PO's interpretation)

Accordingly, in each of these examples, *Kumar's* system is selectively turning off one or more features in a feature vector based on a value of one of the static data points being within a predetermined range— i.e., whether or not the features have any value, or are blank/null. *Dec*., ¶¶104-106. This is consistent with PO's proposed interpretation of the claim language it advanced in parallel litigation. Namely, PO suggested the plain and ordinary meaning of "within a predetermined range" is not limited to "a numerical range or a subset of a 'particular scale' with at least a lower limit or an upper limit, but not necessarily both." Ex. 1007 (*Responsive Markman Brief, Dkt. 304*), 59-60. PO also argued the "assumption that the predetermined range has to be a numerical range is [] inconsistent with the '844 specification." Ex. 1009 (*Responsive Markman Brief, Dkt. 98*), 66-67 (citing overlapping disclosure in the parent '844 Patent concerning the "legal information" example also found in the '869 specification). PO continued that "predetermined ranges are not restricted to numeric values on a spectrum…the 'predetermined range' in the case of 'legal language' in the body of the executable file is that if such language is present, then the learning classifier allows certain other features of the vector to be turned off. Such a determination does not require three numeric data points, but rather the 'predetermined range' in this disclosed instance is whether or not the executable file includes legal information." *Id*., 67-68. In support, PO pointed to the disclosed

example in which the classifier "may detect that the file does not contain 'legal information' in the file." '869 Patent, 7:43-64. "In an example where the learning classifier detects the presence of legal information does not exist in the executable file, this may trigger the learning classifier to… turn off other features related to an evaluation of the 'legal information.'" *Id*. Like this example, *Kumar* detects the absence of, e.g., organization or signature type data, and turns off these features and other related features and sends only the boolean features for classification. *Dec*., ¶¶104-106 (citing *Tahan* and noting it was generally known in the art to turn off features of a feature vector during classification of a file).

b. "within a predetermined range" (District Court's construction)

Petitioner notes that the District Court construed "within a predetermined range" with respect to the parent '844 Patent to carry its plain and ordinary meaning, where "range" requires a "numerical range or a subset of a 'particular scale' with at least a lower limit or an upper limit, but not necessarily both." Ex. 1010 (*Markman Order*), 18. When briefing this language for the '869 Patent, PO acknowledged the court's construction, but expressly preserved its rights to appeal based on the arguments it advanced with respect to the '844 Patent. Ex. 1007 (*Responsive Markman Brief, Dkt. 304*), 59-60. To the extent PO argues that the District Court construction should be interpreted broadly enough to encompass a binary situation

such that the existence or non-existence of a value defines the lower or upper limit, *Kumar* satisfies this interpretation as set forth above.

Alternatively, to the extent the District Court construction requires something more than simply the existence or absence of a value, it would have been obvious to modify *Kumar's* methods to perform this step based on the teachings of *Tahan*, which satisfies this limitation under any construction.

Tahan discloses a similar static malware detection method that "operates by analyzing short segments extracted from the file examined." *Tahan*, 957. "Each segment comprises a number of 3 grams depending on the length of the segment[.]" *Id.* "Three features can be derived for each segment: Spread, MFG, and Entropy." *Id.* These features "are in fact meta-features as they are used to represent *features of features* (features of basic 3-grams)." *Id.* (emphasis omitted).

During a malware detection phase, *Tahan's* method uses each segment's entropy value to selectively turn on or off the associated segment's features (i.e., the 3-grams associated with the segment) based on an evaluation of whether the entropy value is within a predetermined range. Specifically, "[t]he algorithm gets two parameters EntropyLow and EntropyHigh." *Id.*, 958. *Tahan's* system then "check[s] if the entropy is smaller than EntropyLow threshold or entropy is larger than Entropy-High…[and] [i]f so then discard[s] the segment and continue[s] segment iteration." *Id.* In other words, if the segment's entropy value falls outside the range

defined by the EntropyLow and EntropyHigh thresholds, the segment, and the 3-gram features associated with the segment, are discarded. *Dec*., ¶¶110-111.

Pursuant to *Tahan's* teachings, it would have been obvious to a PHOSITA to configure *Kumar's* detection logic to selectively turn on/off features in a feature vector based on numerical values of those features. *Id*., ¶¶107-126. For example, *Kumar* discloses numerical static telemetry data points which are extracted from a file and used as features for the classifier, such as "Version" ("Version number of the binary file"). *Kumar*, [0028], [0032]. *Kumar* also explains its disclosed features are "example features derived from the telemetry data." *Id*. [0028], [0055] ("the features described herein are only examples"). *Kumar* discloses that "the classifier is not limited to these types of features, as others…may be used instead of or in addition to some or all of the ones exemplified herein." *Id*. A PHOSITA would have found it obvious to extract other known numerical features to benefit classification, such as a file size. *Dec*., ¶¶113-114 (citing Ex. 1023 (FIRMA)). Both "version" and "file size" features are "numerical" consistent with the '869 Patent's specification. *See*, '869 Patent, Table 1.1 (listing "linker version," "OS version," "image version," "subsystem version," "file version number" along with "file size" and "code size" under the "Numeric values" category).

Based on the teachings of *Tahan*, it would have been obvious to modify *Kumar's* detection logic to selectively turn on/off these numerical features, such as

the version or file size features, by comparing these values to a threshold, as disclosed in *Tahan*. *Dec*., ¶¶115-118 (describing process for turning on/off "version" and "file size" features). Indeed, it was known in the art that certain numerical features are irrelevant for classification when their values fall below a certain range. *Dec*., ¶¶115-117. Ex. 1023 (FIRMA) discusses that 4 KB is "the minimum size of a working malware" that has been observed. Ex. 1023, 150. With this, a PHOSITA would have understood that file size features extracted by *Kumar's* detection logic with numerical values under 4 KB may be discarded. *Dec*., ¶115. In this way, these features would be "turned off" based on a "numerical range or a subset of a 'particular scale' with at least a lower limit or an upper limit." *Id*. Discarding file size features below 4 KB would reduce redundant processing, thereby optimizing classification over the most relevant features. *Id*.

Additionally, it would have been obvious to a PHOSITA to configure *Kumar's* detection logic to selectively turn on/off features (e.g., a filename) in a feature vector based on the feature's entropy value being within a predetermined range. *Dec*., ¶¶119-126. As described above, *Kumar* discloses extracting static data points from files, including, for example, a "File Name." *Kumar*, [0028] (table listing exemplary features), [0052] (further describing that filenames "can be converted into a set of trigrams where each trigram index represents three consecutive letters in the file name"). *Kumar*, like *Tahan*, also generally describes

deriving meta-features (features of features). *Id*., [0035]. Namely, *Kumar* discloses deriving the "entropy of the observed filenames" to be "used as a feature" for classification, which represents "[s]ome measure of the difference in the file names" across the same file running on different computers. *Id*.; *Dec*., ¶119; *compare*, '869 Patent Table 1.1. (listing "Section Entropy" as a numerical static data feature).

When modified by *Tahan*, *Kumar's* threat detection system would evaluate filename entropy values to determine whether these values fall within a predetermined range—e.g., whether entropy is below the EntropyLow threshold or exceeds the Entropy-High threshold. *Dec*., ¶¶119-120. This evaluation would optimize *Kumar's* classification process, selectively discarding (i.e., turning off) irrelevant filenames with too low or too high entropy values during runtime thereby reducing the number of features considered based on the specific file being evaluated. *Dec*., ¶120. A PHOSITA would have understood "discarded" filenames are "turned off" within the meaning of the '869 Patent. *Dec*., ¶121 (describing '869 Patent, 7:43-64). Indeed, it was known in the art that discarded data falling outside a threshold is not considered during classification. *Dec*., ¶¶122-123 (describing *Davuth* and *Yih*). In operation, a specific file with filename entropy values below EntropyLow would indicate little (or no) difference in filenames from multiple instances of this file, and the classifier would discard this data point as irrelevant to classification. *Dec*., ¶124. Conversely, if entropy values exceed EntropyHigh, this

would indicate too much variability such that the filename logged is indiscernible and not a good measure of data within the file and is similarly discarded (i.e., *turned off*). *Id*.

A PHOSITA would have appreciated a reasonable expectation of success making this modification. *Dec*., ¶¶125-126. *Kumar* already discloses that its detection logic can be enhanced such that it can "use larger resource limits, more signatures, more in-depth processing, and so forth." *Kumar*, [0024]. Adding an additional processing routine would have required only programming changes to *Kumar's* already present detection logic with the tradeoff of optimizing classification on a "file specific" basis. *Dec*., ¶¶125-126.

[1(f)] providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

Kumar discloses using *a support vector machine* to classify the feature vectors generated during malware detection. *Kumar* discloses an exemplary list of classifiers operable with its disclosed threat detection system, including a "support vector machine" (SVM):

> In one implementation of the telemetry classifier, a multiclass classifier (e.g., logistic regression, naïve Bayes, **support vector machine**, neural network, boosted decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like) may be used to assign an unknown telemetry report to a particular malware family, with a generic malware family representing other families, a particular benign group of applications,

> and/or the generic benign class representing any remaining benign applications.

Kumar, [0027], [0039]. As explained above, a PHOSITA would have been motivated to use an SVM as the classifier in *Kumar's* threat detection system to appreciate the known benefits and advantages of using the same compared to other AI-based classifiers. *Supra*, Claim 1(e); *Dec*., ¶¶127-128.

Kumar discloses the output of classification is a probabilistic model that indicates whether the unknown executable file fits into one of the categories of files. "[T]he backend malware detection logic 102 may include the results of a file based malware classifier which **assigns a probability to the file that it belongs to a particular malware family**, the generic malware family, a particular benign family, or the generic benign family." *Kumar*, [0037], [0040]-[0043]. An exemplary probabilistic model is shown in the table below:

Maximum Category Probability	Category Probability	First substring	Second substring
P_{Benign}	>0.9	"Benign_4"	""
P_{Benign}	0.7-0.9	"Benign_3"	(The empty
P_{Benign}	0.5-0.7	"Benign_2"	string)
P_{Benign}	0.333-0.50	"Benign_1"	
$P_{Inconclusive}$	>0.333	"Inconclusive"	
$P_{Malware}$	0.333-0.50	"Malware_1"	The most likely
$P_{Malware}$	0.5-0.7	"Malware_2"	family string
$P_{Malware}$	0.7-0.9	"Malware_3"	(e.g. Rbot)
$P_{Malware}$	>0.9	"Malware_4"	

Id., [0043]. A PHOSITA would have understood or found obvious, based on *Kumar's* disclosures, that the SVM classifier would build a probabilistic model to

categorize the detected executable file. *Dec*., ¶¶95-99, 127-128; *Kumar* [0045] (disclosing above model applicable to several classifiers, including the "support vector machine").

2. Claim 2

2. The computer-implemented method according to claim 1, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Kumar discloses that antimalware client component 116 "extract[s]" features and characteristics (i.e., *plurality of static data points*) from the identified executable file to send to backend service 104 for analysis. *Supra*, Claim 1(d). While *Kumar* discloses it can unpack a file to obtain certain data points (e.g., *Kumar*, [0055] "unpacked strings"), a PHOSITA would have understood or found obvious that *Kumar's* other exemplary extracted features, shown in the table below, are extracted without decryption or unpacking:

Feature	Description
File Name	Name of the PE file
Original File Name	File name in the original report
File Name Matches Original?	Whether the file name in this report matches the file name in the original report
File Type	Type of file it is
Signer Name	Organization that signed the file
Signing Authority	Certificate authority that issued the signature
Signature Type	File signed or not; if signed, whether the signature is legitimate or invalid
Description	Description of the file in the header
Organization	Manufacturer of the binary file
Version	Version number of the binary file
Packed	Whether the file is packed
LS Hash	Locality sensitive hash
Behavior Feature	Represents the simple behavior that caused the report to be generated

Dec., ¶¶129-130 (noting the majority of the information provided in the table above is extracted from PE file headers, which are not packed or encrypted).

First, if the identified executable is not packed and encrypted, *Kumar* would extract features without unpacking or decryption. *Dec*., ¶¶129-130 (noting that *Kumar* can obtain file strings "directly from the file" without unpacking for files that are not packed or encrypted). Indeed, the District Court rejected that an "executable file" needs to be "an encrypted or packed computer file or program[,]" applying the plain and ordinary meaning. Ex. 1010 (*Markman Order*), 17. Therefore, this claim broadly captures classification of unpacked and unencrypted files.

For packed and encrypted files, a PHOSITA would have understood *Kumar* would extract many of the above features without unpacking or decryption. *Dec*.,

¶¶129-130 (noting "[s]tatic features are those obtained by simply investigating the binary contents of a file" whereas "[d]ynamic features require executing the file in a sandbox-type of monitoring system"); *compare* '869 Patent, Table 1.1 (listing same features, such as file name, file type, etc.), 6:45-47 (noting these points can be extracted "regardless of whether or not they are encrypted and/or compressed"). Alternatively, a POSITA would have found it obvious to use a known static feature extraction tool, such as "PE Dump" to obtain PE file features, such as filenames without unpacking or decryption. *Dec*., ¶¶131-132 (describing reasonable expectation of success).

3. Claim 3

3. The computer-implemented method according to claim 1, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

Supra, Claim 1(f) (discussing *Kumar's* SVM building a probabilistic model). *Kumar* discloses that its probabilistic model is built from the following function:

$$s[j,n] = \sum_{f=0}^{F-1} w(f,j)x(f,n) + b(j)W(i,j)$$

Kumar, [0040]. Here, "w(f,j) is the binary classifier weight for feature f for the jth family classifier…[t]here are Ftotal features, and x(f,n) is the fth feature value for the nth unknown file to be evaluated." *Id*., [0040]. The classifier computes 102 "1 versus the rest" linear classifiers (100 specific malware families, 1 generic malware

family, and 1 benign family), then "computes the overall probability for the nth file." *Id*. If the classifier is set to include an "inconclusive" result, three separate output probabilities are used for the three states, "Malware," Benign," and "Inconclusive":

$$P_{benign}(n)=e^{s[Benign,n]}/(\Sigma_k e^{s[k,n]}+1)$$

$$P_{Malware}(n)=\Sigma_{j=Benign} e^{s[j,n]}/(\Sigma_k e^{s[k,n]}+1)$$

$$P_{Inconclusive}(n)=1/(\Sigma_k e^{s[k,n]}+1)$$

Id., [0041]. The results form the probabilistic model above in Claim 1(f). *Supra*, 1(f).

A PHOSITA would have understood that *Kumar's* functions above perform data analysis and pattern recognition on one or more feature vectors (e.g., applying weights to feature "f") to build its probabilistic model. *Dec*., ¶¶133-134. Indeed, it was well-known that SVMs "are supervised learning models associated with learning algorithms and used to **analy[z]e data** and **recogni[z]e patterns**." *Id*.

4. Claim 4

4. The computer-implemented method according to claim 1, wherein the probabilistic model indicates whether the executable file is harmful.

Kumar discloses its probabilistic model indicates whether the executable file is malware (i.e., *harmful*):

Maximum Category Probability	Category Probability	First substring	Second substring
P_{Benign}	>0.9	"Benign_4"	""
P_{Benign}	0.7-0.9	"Benign_3"	(The empty
P_{Benign}	0.5-0.7	"Benign_2"	string)
P_{Benign}	0.333-0.50	"Benign_1"	
$P_{Inconclusive}$	>0.333	"Inconclusive"	
$P_{Malware}$	0.333-0.50	"Malware_1"	The most likely
$P_{Malware}$	0.5-0.7	"Malware_2"	family string
$P_{Malware}$	0.7-0.9	"Malware_3"	(e.g. Rbot)
$P_{Malware}$	>0.9	"Malware_4"	

Kumar, [0043]; *see also id*., [0037] ("the backend malware detection logic may include the results of a **file based malware classifier** which **assigns a probability to the file that it belongs to a particular malware family**").

5. Claim 5

5. The computer-implemented method according to claim 1, wherein the executable file is identified in response to a detected condition.

Kumar discloses the executable file is identified in response to a detected condition, such as when a potential malware file "is received at the customer front-end 112" or when it "attempt[s] to infect the customer machine front-end 112[.]" *Kumar*, [0022], Fig. 1 (circled numeral one (1)), Fig. 5 ("Receive unknown file"). The detected condition can also be a probing attempt. *Id*., [0049]-[0050].

6. Claim 6

6. The computer-implemented method according to claim 5, wherein the detected condition is user request for a file download.

Kumar discloses identifying an executable file in response to receiving the file at a customer machine and other detected conditions. *Supra*, Claim 5. It would have been obvious to a PHOSITA for the executable file to be identified prior to receipt when the user requests download. *Dec*., ¶¶135-140. *Kumar* already generally discloses performing a threat detection analysis "before downloading the file to the customer's machine 112" and analyzing "system API calls." *Id*., [0036], [0039]. A PHOSITA therefore would have found it obvious to identify the file prior to download, such as when an API call requesting download is made. *Dec*., ¶¶135-140 (further describing use of router or "proxy" to detect malware before download and motivations and reasonable expectation of success related thereto).

7. Claim 7

7. The computer-implemented method according to claim 5, wherein the detected condition is the detection of a new file attempting to execute.

Supra, Claim 5 (file identified when it "attempt[s] to infect the customer machine front-end 112"). A PHOSITA would have understood that to infect the customer machine, the file must attempt to execute. *Dec*., ¶141.

8. Claim 8

8. The computer-implemented method according to claim 1, wherein the plurality of static data points represent predefined character strings in the executable file.

The '869 Patent does not use the term "predefined character strings" outside the claims. However, the '869 Patent generally provides exemplary static data points falling within the "Strings/Byte sequences" category, shown below:

TABLE 1.1

Numeric values	Nominal values	Strings/Byte sequences	Boolean values
File size	initialize	Comments	Address Of Entry Point Anomaly
linker version	Un-initialize	company name	Image Base Anomaly
code size	entry point	file description	Section Alignment Anomaly
OS version	subsystem	internal name	Size Of Code Mismatch Anomaly
image version	file subtype	legal copyright	Low Import Count Anomaly
subsystem version	language	original file	Entry Point Anomaly
file version number	file flags masks	private build	certificate Validity
product version number	file flags	product name	Certificate Exception
size of heapr	file OS	special build	Code Characteristics Anomaly
size of stackr	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate Subject	Number of Functions Anomaly
	Earliest Data Byte	Imports	Function Name Anomaly
	resources	Exports	PE Header Anomaly
	resources language	Section Names	High Section Count Anomaly
	resource Encoding	Non-resource section strings	PE Magic Validity
	resource code page		Resource Exception
	resource size		VR Code Ratio Anomaly
	DLL characteristics		Import Exception

'869 Patent, Table 1.1. This "string sequence data" may be "processed as n-grams[.]" *Id*., 8:15-23. And Patent Owner has explained "'character strings' can be numerical values…used interchangeably with 'byte sequences'[.]" Ex. 1009 (*Responsive Markman Brief, Dkt. 98*), 66 (citing parent '844 Patent, 8:29-48, "'n-grams' made up of 'string and/or byte sequences'").

Like the ’869 Patent, *Kumar* discloses static data points that fall within this “Strings/Byte sequences” category, such as “File Name,” “Version,” “Certificate information,” etc. *Kumar*, [0028]; *supra*, Claim 1(a) (showing table of static data points), Claim 1(e)(i). Further, like the ’869 Patent, this data can be represented as 3-byte n-grams (trigrams). *Id*., [0052]. Therefore, *Kumar* discloses static data points as *character strings*. *Dec*., ¶¶142-146.

Moreover, these character strings are “predefined” under the intrinsic record. During prosecution of the parent ’844 Patent, Applicant proposed several amendments clarifying that “static data points” correspond to “points of reference (in the file content)” such as to “particular sections or portions of the file.” ’844 Parent FH, 134-135. The Examiner found that the prior art—Wojnowicz—disclosed “a string of hexadecimal characters” and that because an example static data point is “string information[,”] the prior art disclosed this concept. Applicant ultimately removed the “points of reference” language and added “*the plurality of static data points represent predefined character strings in the executable file*[;]” suggesting that Wojnowicz’s “fixed-length file chunks do not represent predefined character strings” where “each fixed-length file chunk is, itself, a random (e.g. undefined) character string having no initial significance to the system of Wojnowicz.” *Id*., 312. Here, *Kumar’s* filename, version, certificate information, etc. character strings correspond directly to particular predefined sections of the file—i.e., the file’s

header. *Dec*., ¶¶142-146. These strings therefore are (pre)defined and are significant to classification. *Id*.

9. Claim 9

9. The computer-implemented method according to claim 1, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

Kumar discloses its detection logic is "updateable" such that it "may be updated based upon one or more updating criteria, such as…new information and/or the like." *Kumar*, [0006]. In this regard, *Kumar* discloses "**retraining** and/or replacing the logic[.]" *Id*., [0047]. It would have been obvious to a PHOSITA to facilitate retraining in *Kumar* using a well-known feedback approach where the determination of whether the executable file is malware (i.e., harmful) is fed back into the system as updated training data. *Dec*., ¶¶147-150. Feeding back a malware determination would provide new "signature data" to the detection logic, which *Kumar* explains is "needed to remediate the problem." *Id*.; *Kumar*, [0024]-[0025].

10. Claim 10

10[Pre]. A system comprising:

Kumar discloses an exemplary computing system which implements the various disclosed embodiments:

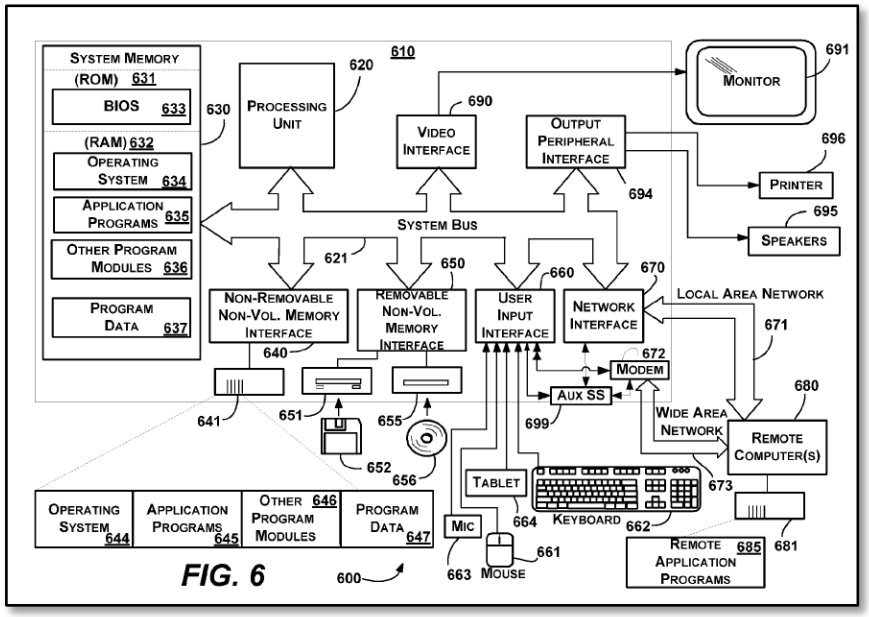


FIG. 6

Kumar, Fig. 6; *see also id*., [0075] ("The invention may be described in the general context of computer-executable instructions, such as program modules, being executed by a computer"); *supra*, Claim 1[Pre].

[10(a)] at least one memory; and

Kumar's system includes a "system memory 630." *Kumar*, [0076], Fig. 6.

[10(b)] at least one processor operatively connected with the memory and configured to perform operation of:

Kumar discloses its "processing unit 620" is operatively connected to system memory 630." *Kumar*, [0076], Fig. 6. *Kumar's* method is "performed at least in part on at least one processor[.]" *Id*., claim 1.

[10(c)] identifying static data points that may be indicative of either a harmful or benign executable file;

Supra, Claim 1(a).

[10(d)] associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

Supra, Claim 1(b).

[10(e)] identifying an executable file to be evaluated;

Supra, Claim 1(c).

[10(f)] extracting a plurality of static data points from the identified executable file;

Supra, Claim 1(d).

[10(g)] and generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

Supra, Claim 1(e).

[10(g)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

Supra, Claim 1(e)(i).

[10(h)] providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

Supra, Claim 1(f).

11. Claim 11

11. The system according to claim 10, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Supra, Claim 2.

12. Claim 12

12. The system according to claim 10, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

Supra, Claim 3.

13. Claim 13

13. The system according to claim 10, wherein the probabilistic model indicates whether the executable file is harmful.

Supra, Claim 4.

14. Claim 14

14. The system according to claim 10, wherein the plurality of static data points represent predefined character strings in the executable file.

Supra, Claim 8.

15. Claim 15

15[Pre]. A computer-readable storage device containing instructions, that when executed on at least one processor, causing the processor to execute a process comprising:

Supra, Claim 10[Pre]; *Kumar*, [0077], [0080], claim 1, 17.

[15(a)] identifying static data points that may be indicative of either a harmful or benign executable file;

Supra, Claim 1(a).

[15(b)] associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

Supra, Claim 1(b).

[15(c)] identifying an executable file to be evaluated;

Supra, Claim 1(c).

[15(d)] extracting a plurality of static data points from the identified executable file;

Supra, Claim 1(d).

[15(e)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

Supra, Claim 1(e).

[15(e)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

Supra, Claim 1(e)(i).

[15(f)] providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

Supra, Claim 1(f).

16. Claim 16

16. The computer-readable storage device according to claim 15, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Supra, Claim 2.

17. Claim 17

17. The computer-readable storage device according to claim 15, wherein the plurality of static data points represent predefined character strings in the executable file.

Supra, Claim 8.

18. Claim 18

18[Pre]. A computer-implemented method comprising:

Supra, Claim 1[Pre].

[18(a)] identifying static data points that may be indicative of either a harmful or benign executable file;

Supra, Claim 1(a).

[18(b)] associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

Supra, Claim 1(b).

[18(c)] identifying an executable file to be evaluated;

Supra, Claim 1(c).

[18(d)] extracting a plurality of static data points from the executable file;

Supra, Claim 1(d).

[18(e)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

Supra, Claim 1(e).

[18(e)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

Supra, Claim 1(e)(i).

[18(f)] evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

Supra, Claim 1(f). *Kumar* also discloses various other machine learning models (*Kumar*, [0027], [0039]) which a PHOSITA would substitute interchangeably depending on desired classification parameters. *Dec*., ¶¶151-153.

19. Claim 19

19. The computer-implemented method according to claim 18, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Supra, Claim 2.

20. Claim 20

20. The computer-implemented method according to claim 18, wherein the machine learning model comprises an artificial neural network.

Supra, Claim 18(f) (discussing interchangeability between different models). *Kumar* discloses a "neural network" as its machine learning model. *Kumar*, [0027], [0039]; *Dec*., ¶¶154-155.

21. Claim 21

21. The computer-implemented method according to claim 18, wherein the machine learning model comprises a support vector machine.

Supra, Claim 1(f).

22. Claim 22

22. The computer-implemented method according to claim 18, wherein the machine learning model comprises a machine learning decision tree.

Supra, Claim 18(f) (discussing interchangeability between different models). *Kumar* discloses a "boosted decision tree" as its machine learning model. *Kumar*, [0027], [0039]; *Dec*., ¶156.

23. Claim 23

23. The computer-implemented method according to claim 18, wherein the machine learning model comprises a Bayesian network.

Supra, Claim 18(f) (discussing interchangeability between different models). *Kumar* discloses "naïve Bayes" as its machine learning model. *Kumar*, [0027], [0039]; *Dec*., ¶157.

24. Claim 24

24. The computer-implemented method according to claim 18, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

PO has stated the plain and ordinary meaning of "support vector processing" "is simply 'processing that involves a support vector.'" Ex. 1009 (*Responsive Markman Brief, Dkt. 98*), 69 (noting these "can be processed both by support vector machines and by other machine learning classifiers, such as neural networks"). Petitioner asks the Board to adopt PO's interpretation. *Kumar* discloses machine learning models, such as "support vector machines" and "neural networks" for classification (*supra*, Claim 1(f), 18(f), 20-23), which a PHOSITA would have understood involves *support vector processing*. *Dec*., ¶¶158-159.

25. Claim 25

25. The computer-implemented method according to claim 18, wherein the machine learning model comprises a clustering model.

Kumar discloses several exemplary machine learning models, including "logistic regression, naive Bayes, support vector machine, neural network, boosted

decision tree, other boosted algorithms, ensemble classifiers, hierarchical classifiers and/or the like." *Kumar*, [0027], [0039]. A PHOSITA would have understood or found obvious that several of models use clustering, such as "ensemble classifiers" and "hierarchical classifiers." *Dec*., ¶¶160-162. Alternatively, it would have been obvious to modify *Kumar's* support vector machine to implement well-known clustering techniques or to employ a well-known clustering model as its classifier. *Dec*., ¶¶163-165 (describing reasonable expectation of success). Indeed, it was known in the art to implement clustering algorithms in threat detection systems, including those with labeled datasets. *Id*. (citing *Tahan* and references discussed therein).

26. Claim 26

26. The computer-implemented method according to claim 18, wherein the executable file is identified in response to a detected condition.

Supra, Claim 5.

27. Claim 27

27. The computer-implemented method according to claim 26, wherein the detected condition is user request for a file download.

Supra, Claim 6.

28. Claim 28

28. The computer-implemented method according to claim 26, wherein the detected condition is the detection of a new file attempting to execute.

Supra, Claim 7.

29. Claim 29

29. The computer-implemented method according to claim 18, wherein the plurality of static data points represent predefined character strings in the executable file.

Supra, Claim 8.

30. Claim 30

30. The computer-implemented method according to claim 18, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

Supra, Claim 9.

31. Claim 31

31. The computer-implemented method according to claim 18, further comprising preventing execution of the executable file based on classifying the executable file as harmful.

Kumar discloses its threat detection system processes "potential malware" "before being allowed to run." *Kumar*, [0022]. After a file is identified as malware (i.e., *harmful*), "remedial action" is taken such as to "clean delete or quarantine the file containing the malware[.]" *Id.*, [0025], Fig. 1 (circle (4) "remediates/blocks"), Fig. 5 (step 534 "Remediate"). A PHOSITA would have understood deleting, blocking, or quarantining a file would prevent its execution. *Dec*., ¶166.

32. Claim 32

32[Pre]. A computer-readable storage device containing instructions, that when executed on a processor, cause the processor to execute a process comprising:

Supra, Claim 15[Pre].

[32(a)] identifying static data points that may be indicative of either a harmful or benign executable file;

Supra, Claim 1(a).

[32(b)] associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

Supra, Claim 1(b).

[32(c)] identifying an executable file to be evaluated;

Supra, Claim 1(c).

[32(d)] extracting a plurality of static data points from the executable file;

Supra, Claim 1(d).

[32(e)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

Supra, Claim 1(e).

[32(e)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

Supra, Claim 1(e)(i).

[32(f)] evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

Supra, Claim 1(f), 18(f).

33. Claim 33

33. The computer-readable storage device according to claim 32, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Supra, Claim 2.

34. Claim 34

34. The computer-readable storage device according to claim 32, wherein the machine learning model comprises an artificial neural network.

Supra, Claim 20.

35. Claim 35

35. The computer-readable storage device according to claim 32, wherein the machine learning model comprises a support vector machine.

Supra, Claim 1(f), 21.

36. Claim 36

36. The computer-readable storage device according to claim 32, wherein the machine learning model comprises a machine learning decision tree.

Supra, Claim 22.

37. Claim 37

37. The computer-readable storage device according to claim 32, wherein the machine learning model comprises a Bayesian network.

Supra, Claim 23.

38. Claim 38

38. The computer-readable storage device according to claim 32, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

Supra, Claim 24.

39. Claim 39

39. The computer-readable storage device according to claim 32, wherein the machine learning model comprises a clustering model.

Supra, Claim 25.

40. Claim 40

40. The computer-readable storage device according to claim 32, wherein the executable file is identified in response to a detected condition.

Supra, Claim 5.

41. Claim 41

41. The computer-readable storage device according to claim 40, wherein the detected condition is user request for a file download.

Supra, Claim 6.

42. Claim 42

42. The computer-readable storage device according to claim 40, wherein the detected condition is the detection of a new file attempting to execute.

Supra, Claim 7.

43. Claim 43

43. The computer-readable storage device according to claim 32, wherein the plurality of static data points represent predefined character strings in the executable file.

Supra, Claim 8.

44. Claim 44

44. The computer-readable storage device according to claim 32, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

Supra, Claim 9.

45. Claim 45

45. The computer-readable storage device according to claim 32, wherein the process further comprises preventing execution of the executable file based on classifying the executable file as harmful.

Supra, Claim 31.

46. Claim 46

46[Pre]. A system comprising:

Supra, Claim 10[Pre].

[46(a)] at least one memory;

Supra, Claim 10(a).

[46(b)] at least one processor configured to perform operation of:

Supra, Claim 10(b).

[46(c)] identifying static data points that may be indicative of either a harmful or benign executable file; and

Supra, Claim 1(a).

[46(d)] associating the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files; and

Supra, Claim 1(b).

[46(e)] at least one processor configured to perform operation of:

Supra, Claim 10(b). To the extent PO suggests this claim requires different processors, a PHOSITA would have understood or found obvious that *Kumar* discloses multiple processors performing its claimed methods. *Dec*., ¶167; *Kumar*, claim 1 ("a method performed at least in part **on at least one processor**…"), [0073]-

[0074], [0085] ("multiple processing chips or multiple devices can share the performance of one or more functions described herein").

[46(f)] identifying an executable file to be evaluated;

Supra, Claim 1(c).

[46(g)] extracting a plurality of static data points from the executable file;

Supra, Claim 1(d).

[46(h)] generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files,

Supra, Claim 1(e).

[46(h)(i)] wherein one or more features of the feature vector are selectively turned on or off based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

Supra, Claim 1(e)(i).

[46(j)] evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.

Supra, Claim 1(f), 18(f).

47. Claim 47

47. The system according to claim 46, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Supra, Claim 2.

48. Claim 48

48. The system according to claim 46, wherein the machine learning model comprises an artificial neural network.

Supra, Claim 20.

49. Claim 49

49. The system according to claim 46, wherein the machine learning model comprises a support vector machine.

Supra, Claim 1(f), 21.

50. Claim 50

50. The system according to claim 46, wherein the machine learning model comprises a machine learning decision tree.

Supra, Claim 22.

51. Claim 51

51. The system according to claim 46, wherein the machine learning model comprises a Bayesian network.

Supra, Claim 23.

52. Claim 52

52. The system according to claim 46, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

Supra, Claim 24.

53. Claim 53

53. The system according to claim 46, wherein the machine learning model comprises a clustering model.

Supra, Claim 25.

54. Claim 54

54. The system according to claim 46, wherein the executable file is identified in response to a detected condition.

Supra, Claim 5.

55. Claim 55

55. The system according to claim 54, wherein the detected condition is user request for a file download.

Supra, Claim 6.

56. Claim 56

56. The system according to claim 54, wherein the detected condition is the detection of a new file attempting to execute.

Supra, Claim 7.

57. Claim 57

57. The system according to claim 46, wherein the plurality of static data points represent predefined character strings in the executable file.

Supra, Claim 8.

58. Claim 58

58. The system according to claim 46, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

Supra, Claim 9.

59. Claim 59

59. The system according to claim 46, further comprising at least one processor configured to perform operation of: preventing execution of the executable file based on classifying the executable file as harmful.

Supra, Claim 31, 46(e) (functions can be shared between multiple processors).

V. DISCRECTIONARY DENIAL IS NOT WARRANTED

A. The *General Plastic* Factors Favor Institution

The Board set forth seven factors it considers in determining whether to exercise discretion to deny a petition under § 314(a). *General Plastic Industrial Co., Ltd. v. Canon Kabushiki Kaisha*, IPR2016-01357, Paper 19 at 15-19 (September 6, 2017) (precedential). Here, these factors weigh heavily in favor of institution.

The ’869 Patent has been challenged in a prior PGR petition by Sophos Inc., PGR2023-00031 (“Sophos PGR”). There, Sophos presented six grounds of unpatentability, including three grounds relating to §112 indefiniteness and written description issues. In the three prior art grounds, Sophos relied on Ideses and Mori alone or in combination—neither of which are relied upon in the instant challenge. The proposed grounds in the instant petition differ substantially from those in Sophos’s petition, relying on non-cumulative art in a non-cumulative manner. Further, Sophos’s petition was prepared independently of Petitioner. Petitioner was not involved in the Sophos IPR and only learned of it after it was filed. Nor did Sophos contribute to the instant petition in any way. Additionally, Sophos’s petition was just recently filed such that Patent Owner has not filed its preliminary response and no institution decision has issued.

Additionally, just days ago, the ’869 Patent was challenged in a prior IPR petition by AO Kaspersky Lab, IPR2023-01160 (“Kaspersky IPR”). Kaspersky

presents two grounds relying similarly on Ideses or Ideses and a new secondary, Sharma, challenging claims 19, 20, 22, 24-34, 36, 38-48, 50, 52-59. For the same reasons Petitioner's challenge is non-cumulative of the Sophos PGR, the instant petition is non-cumulative of the Kaspersky IPR.

Given the independent nature of the instant IPR, the non-cumulative grounds presented herein, and impossibility of road-mapping concerns, the Board should decline to exercise its discretion under § 314(a).

1. Factor 1—Whether the same petitioner previously filed a petition directed to the same claims of the same patent

That Petitioner is not a party to any prior proceeding and that this is Petitioner's first challenge to the '869 Patent "weighs especially heavily against a discretionary denial." *Unified Patents, Inc. v. Certified Measurement, LLC*, IPR2018-00548, Paper No. 7 at 7-8 (Sep. 5, 2018). "[A]pplication of the General Plastic factors is not limited solely to instances when multiple petitions are filed by the same petitioner." *Valve Corp. v. Electronic Scripting Products, Inc.*, IPR2019-00062, Paper No. 11 at 2 (Apr. 2, 2019). In *Valve*, however, the Board found that petitioner (Valve) had a "significant relationship" with previous petitioner (HTC) and filed its petition only after HTC's petition had been denied by the Board. *Id*., 9-10, 12-13. Petitioner neither had, nor has, any "significant relationship" with any prior Petitioner here. Petitioner is not coordinating with Sophos or AO Kaspersky on any petitions for IPR, there is no business relationship between Sophos/AO

Kaspersky and Petitioner, and there is no overlap in accused products. *See Foursquare Labs, Inc. v. Mimzi LLC*, IPR2019-01287, Paper 11 (Jan. 14, 2020) (petitioners individually sued with no evidence of coordination weighed against denying institution).

2. Factors 2-5

The second to fifth factors "bear little relevance" here because Petitioner has never before challenged the '869 Patent. *Unified Patents* at 7-8 ("Once resolution of factor 1 indicates that Petitioner had not previously filed a petition against the same patent, factors 2-5 bear little relevance unless there is evidence in the record of extenuating circumstances such as a showing that there was a previous petitioner who challenged the same patent and that the previous petitioner and the current petitioner planned the staggered petitions.") (internal citation omitted); *Qualcomm Inc. v. Monterey Research, LLC*, IPR2020-01493, Paper 11 at 17 (PTAB Mar. 8, 2021). No such extenuating circumstances are present here. Nor is there a possibility of "road-mapping" from an earlier preliminary response or Board decision. The Sophos PGR petition was filed on May 8th and afforded a filing date on June 6, 2023. Ex. 1011. Accordingly, PO's preliminary response is not due until September and a decision on institution is expected in December. Similarly, the Kaspersky IPR was just filed on June 30th, 2023, and no filing date has been afforded yet. A POPR would not be expected until October and an institution decision in January.

3. Factor 6 —The finite resources of the Board

"The sixth and seventh factors are efficiency considerations." *Valve*, IPR2019-00062, Paper 11 at 15. The instant Petition presents non-cumulative Grounds with new art. Sophos presented six grounds total, including three prior art grounds relying on Ideses, Mori, or the combination of the two. And Kaspersky presented two grounds also relying on Ideses challenging a subset of the claims. Petitioner presents a single ground challenging all claims 1-59 in view of new art. Petitioner's combination is fundamentally different than Ideses, Mori, Sharma, or their combination. As explained above, *Kumar* and *Tahan's* systems/methods analyze static data points, assign entropy values to certain data points (e.g., filenames), and turn on/off features in a feature vector representing these static data points during evaluation based on their entropy values compared to an entropy threshold. *Supra*, Sec. IV.A. In contrast, in Ideses, Mori, Ideses in view of Mori, or Ideses in view of Sharma, to turn on/off features in a feature vector, these references utilize a feature vector generation process involving "pruning" insignificant features or a word extracting unit to extract words, create a feature vector from the extracted words, and assign zero values to words that do not appear in a predefined dictionary of terms. Ex. 1011 (Sophos PGR Petition), 33-35, 66-69; Ex. 1012 (Kaspersky IPR Petition).

Petitioner's grounds therefore rely on non-cumulative art with fundamentally different disclosures.

4. Factor 7 —The requirement under 35 U.S.C. § 316(a)(11) to issue a final determination not later than 1 year after the date on which the Director notices institution of review

There is no readily identifiable roadblock for the Board to issue a final determination within the statutory one-year limit.

Accordingly, the Board should reach the merits of this petition, and institute review of all Challenged Claims.

B. The *Advanced Bionics* Framework Favors Institution

Under the two-part *Advanced Bionics* framework applying the *Becton, Dickinson* factors, the Board should not exercise its discretion to deny institution under 35 U.S.C. § 325(d). Part one considers whether the same or substantially the same prior art or arguments were previously presented to the office. *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).

None of the references asserted herein were cited during prosecution of the '869 Patent, so the Examiner did not consider the same art presented herein. *See generally* '869 FH. During prosecution of the parent '844 Patent, the Examiner considered a distinct "entropy" based reference—Wojnowicz. *Supra*, '844 Parent

Prosecution Summary (describing Wojnowicz). There, to differentiate Wojnowicz, Applicant added "*the plurality of static data points represent predefined character strings in the executable file*" to the independent claims and argued that Wojnowicz's "fixed-length file chunks do not represent predefined character strings" where "each fixed-length file chunk is, itself, a random (e.g. undefined) character string having no initial significance to the system of Wojnowicz." '844 Parent FH, 312.

Here, the independent claims do not recite this language. This concept is claimed in dependent claims 8, 14, 17, 29, 43, and 57. For these claims, in contrast to the Examiner's rejection in view of Wojnowicz, Petitioner relies on *Kumar's* teachings of static data points representing predefined character strings, e.g., *Kumar's* filenames. Regarding selectively turning on/off features based on values of these predefined character strings representing filenames, Petitioner advances *Tahan's* entropy process in combination with *Kumar's* teachings such that entropy values associated with *Kumar's* predefined character string features (e.g., filenames) determine whether those features are discarded (i.e., turned off) or evaluated. Therefore, Petitioner's art is non-cumulative to the art applied during prosecution of the parent '844 Patent.

C. 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.*, IPR2020-00019, Paper 11 at 2–3, 6 (PTAB Mar. 20, 2020) (precedential). Those factors favor institution here.

1. 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).

2. 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). Understanding "scheduled trial dates are unreliable

and often change[,]" recent Interim *Fintiv* Guidance[3] dictates "[a] court's scheduled trial date…is not by itself a good indicator of whether the district court trial will occur before the statutory deadline for a [FWD]." *Samsung Elec.'s Co., Ltd. et al. v. Staton Techniya, LLC*, IPR2022-00302, Paper 13, at 10 (PTAB July 11, 2022). When analyzing this factor, "the parties [may] rely on time-to-trial statistics" along with "additional supporting factors such as the number of cases before the judge in the parallel litigation and the speed and availability of other case dispositions." *Id*., 10-12 ("In weighing factor two, we follow the Interim *Fintiv* Guidance").

Based on the most recent WDTX statistics, the "actual trial date" is likely to be held mid-January, 2025—28.3 months from September 8, 2022, the date the first Amended Complaint was filed asserting the '869 Patent against CrowdStrike.[4] While the District Court entered a schedule with an August 19, 2024 trial date assigned to each defendant in five separate lawsuits that have been consolidated for pretrial issues, PO has admitted that "median time-to-trial statistics are more

[3] https://www.uspto.gov/sites/default/files/documents/interim_proc_discretionary_denials_aia_parallel_district_court_litigation_memo_20220621_.pdf ("Interim *Fintiv* Guidance").

[4] https://www.uscourts.gov/sites/default/files/fcms_na_distprofile0630.2022_0.pdf (June 30, 2022, 28.3 months for median time-to-trial in WDTX).

appropriate" for consideration under this factor. IPR2023-00289, Paper 6, at 13; *Amendment to Scheduling Order* (Dkt. 252, Ex. 1013), 8. Either way, because the court's consolidation is limited to pretrial issues, separate trials will be required, and trial cannot proceed on August 19, 2024 for all defendants.

The Board's FWD deadline in this case is expected no later than January 2025, likely before or commensurate with when a trial would occur based on the time-to-trial statistics this Board should apply to its analysis. However, "additional supporting factors" are at play weighing towards a longer than average filing-to-trial window. First, the District Court litigation involves thirty-nine total patents, including twelve asserted by PO against Petitioner and two counterclaim infringement allegations brought by Petitioner. As PO previously quoted, U.S. Magistrate Judge Gilliland explained in a December 9, 2022 hearing that this is "a phenomenal case that I'm hoping one day maybe people will write songs about because it is about as complicated as you're going to get with patent cases." IPR2023-00289, Paper 6, at 2.

To accommodate the significant number of allegations, the Court scheduled three Markman hearings, the first was conducted on March 7, 2023, the second was recently held on May 3, 2023 relating to the Counterclaim patents, and the final Markman is scheduled for July 25, 2023, which will involve the Challenged Patent. Ex. 1014 (Dkt. 350). Accordingly, with multiple Markman hearings and a multitude

of claims involved in a phenomenally complex case, the "actual trial date" is much more likely to be delayed until later, commensurate with the median time to trial (or later) and after FWD here.

Moreover, the '869 Patent was added to the litigation in September 2022 as one of six additional asserted patents, introduced over six months after the original six patents were asserted against Petitioner. The '869 Patent, and other late asserted patents, were shoehorned into an already ongoing litigation, resulting in a shortened window to trial. Petitioner acted diligently in filing its Petition against this newly asserted patent, and any shortened window to trial for these patents is the result of PO's own litigation strategy.

In sum, the Board's FWD deadline in this case falls either **before** or commensurate with when a trial would occur based on WDTX time-to-trial statistics. But the additional complicating factors at issue here suggest trial will occur even later in time than the statistics suggest, weighing heavily in favor of institution.

3. Investment in Parallel Proceeding

Beyond claim construction, the parties have invested little in the parallel proceeding. The first and second *Markman* hearings were just recently conducted on March 7, 2023 and May 3, 2023, respectively. The final *Markman* is not scheduled until July 25, 2023. Fact discovery just opened on April 18, 2023 and is not set to conclude until January 18, 2024. Expert discovery does not close until March 12,

2024 and dispositive motions are not due until May 7, 2024. This factor weighs in favor of institution.

4. Overlap

Petitioner stipulates that, if the Board institutes *inter partes* review, Petitioner will not seek resolution in the District Court of any ground of invalidity for the '869 Patent that utilizes *Kumar* or *Tahan*—the prior art references relied upon in the grounds of the instant petition. Petitioner's proposal is comprehensive and addresses the concern of narrowing the issues, as this stipulation ensures there is no overlap between the '869 Patent invalidity theories before the Board and at issue in 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.

5. Same Party

The Parties are the same in the district court litigation. However, members of the Board have noted *Fintiv* addresses only the scenario in which the petitioner is unrelated to a defendant in a parallel proceeding, finding this should weigh against denying institution, but that *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) (noting that 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").

6. 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 presented above weighs strongly in favor of institution. *Id*. (refusing to deny institution under the *Fintiv* framework based solely on the strength of Petitioner's grounds).

VI. CONCLUSION

For the forgoing reasons, Petitioners respectfully request *inter partes* review of the Challenged Claims.

Respectfully submitted,

/s/ Adam P. Seitz
ERISE IP, P.A.
Adam P. Seitz, Reg. No. 52,206
Paul R. Hart, Reg. No. 59,646

COUNSEL FOR PETITIONER

VII. 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 '869 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 Holdings, Inc. et al.*, Case No. 6:22-cv-00241. Petitioner understands the '869 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);
- *Webroot, Inc. et al. v. AO Kaspersky Lab*, Case No. 6:22-cv-00243 (WDTX);
- *Webroot, Inc. et al. v. Forcepoint LLC*, Case No. 6:22-cv-002342 (WDTX);
- *Sophos, Inc. et al. v. Webroot, Inc.*, PGR2023-00031 (PTAB);
- *AO Kaspersky Lab v. Webroot, Inc.*, IPR2023-01160 (PTAB).

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 PTAB@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 PTAB@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

APPENDIX OF EXHIBITS

EX1001	U.S. Patent No. 11,409,869 ("'869 *Patent*")
EX1002	File History of U.S. Patent No. 11,409,869 ("'869 *FH*")
EX1003	Declaration of Markus Jakobsson, Ph.D. ("*Dec*.")
EX1004	U.S. Patent No. 2012/0317644 to Kumar et al. ("*Kumar*")
EX1005	Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features by Gil Tahan et al. ("*Tahan*")
EX1006	File History of U.S. Patent No. 10,599,844 ("'844 *FH*")
EX1007	22-cv-00243, Dkt. 304, Plaintiffs' Responsive Claim Construction Brief ("*Responsive Markman Brief, Dkt. 304*")
EX1008	Declaration of June Ann Munford ("*Munford Dec.*")
EX1009	22-cv-00243, Dkt. 98, Plaintiffs' Responsive Claim Construction Brief ("*Responsive Markman Brief, Dkt. 98*")
EX1010	22-cv-00243, Dkt. 236 Claim Construction Order ("*Markman Order*")
EX1011	*Sophos Inc. v. Open Text Inc.*, PGR2023-00031, Paper 1
EX1012	AO Kaspersky Lab v. Webroot, Inc., IPR2023-01160, Paper 1
EX1013	22-cv-00243, Dkt. 252, Order Granting Fifth Amended Scheduling Order ("*Amendment to Scheduling Order*")
EX1014	22-cv-00243, Dkt. 350, Order Resetting Markman Hearing ("*Dkt. 350*")
EX1015	Declaration of Dr. Ingrid Hsieg Yee ("*Dr. Yee Dec.*")
EX1016	U.S. Patent Application No. 2013/0326625 to Anderson et al. ("*Anderson*")
EX1017	U.S. Patent No. 8,266,698 ("*Seshardi*")
EX1018	U.S. Patent No. 9,489,514 to Mankin et al. ("*Mankin*")
EX1019	Intrusion Detection Using The Support Vector Machine Enhanced With A Feature--Weight Kernel by Songlun Zhao et al. ("*Zhao*")
EX1020	Intrusion Detection Using Neural Networks and Support Vector Machines by Srinivas Mukkamala et al. ("*Mukkamala*")
EX1021	Classification of Malicious Domain Names using Support Vector Machine and Bi-gram Method by Nhauo Davuth et al. ("*Davuth*")
EX1022	Malware behavioral detection and vaccine development by using a support vector model classifier by Ping Wang et al. ("*Wang*")

EX1023	FIRMA: Malware Clustering and Network Signature Generation with Mixed Network Behaviors by M. Zubair Rafique ("*FIRMA*")
EX1024	U.S. Patent Publication No. 2008/0319932 to Wen-tau Yih et al. ("*Yih*")
EX1025	Comparative Analysis of Combinations of Dimension Reduction and Data Mining Techniques for Malware Detection, by Proceso L. Fernandez Jr. et al. ("*Fernandez*")
EX1026	U.S. Patent No. 8,418,249 to Nucci et al. ("*Nucci*")
EX1027	U.S. Patent No. 8,161,548 to Wan ("*Wan*")
EX1028	U.S. Patent No. 8,875,289 B2 to Mahaffey et al. ("*Mahaffey*")
EX1029	U.S. Patent No. 9,349,103 to Eberhardt, III et al. ("*Eberhardt*")
EX1030	U.S. Patent No. 9,043,894 to Dennison et al. ("*Dennison*")
EX1031	U.S. Patent No. 9,306,971 to Altman et al. ("*Altman*")
EX1032	U.S. Patent Publication No. 2007/0245420 to Yong et al. ("*Yong*")
EX1033	U.S. Patent No. 9,721,212 to Gupta et al. ("*Gupta*")
EX1034	U.S. Patent No. 9,324,034 to Gupta et al. ("*Gupta II*")
EX1035	U.S. Patent No. 10,783,254 to Sharma et al. ("*Sharma*")
EX1036	"Ensemble Methods for Malware Diagnosis Based on One-class SVMs" by Xing An ("*Xing*")
EX1037	U.S. Patent No. 9,306,966 to Eskin et al. ("*Eskin*")

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 13,992 words according to the word-processing program used to prepare this document (Microsoft Word).

Dated: July 3, 2023

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 July 3, 2023, a complete and entire copy of this Petition for *Inter Partes* Review and Exhibits were provided via Federal Express to the Patent Owner by serving the correspondence address of record for the '869 patent:

Sprinkle IP Law Group
1301 W. 25th Street, Suite 400
Austin, TX 78705

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