

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SOPHOS LTD. and SOPHOS INC.,
Petitioner

v.

WEBROOT INC.,
Patent Owner.

IPR2023-00528
Patent 10,599,844 B2

Before PATRICK M. BOUCHER, MICHELLE N. WORMMEESTER, and
AARON W. MOORE, *Administrative Patent Judges*.

MOORE, *Administrative Patent Judge*.

DECISION
Denying Institution of *Inter Partes* Review
35 U.S.C. § 314

I. INTRODUCTION

A. *Background*

Sophos Ltd. and Sophos Inc. (together, “Petitioner”) filed a Petition requesting *inter partes* review of claims 1–30 of U.S. Patent No. 10,599,844 B2 (Ex. 1001, “the ’844 patent”). Paper 1 (“Pet.”). Open Text Inc. (“Patent Owner”) filed a Preliminary Response. Paper 13 (“Prelim. Resp.”).

We may institute an *inter partes* review if “the information presented in the petition . . . and any response . . . shows that there is a reasonable likelihood that the petitioner would prevail with respect to at least 1 of the claims challenged in the petition.” 35 U.S.C. § 314(a) (2018).

For the reasons provided below, we determine that Petitioner has not demonstrated a reasonable likelihood that it will prevail in showing the unpatentability of at least one challenged claim. Accordingly, we do not institute *inter partes* review.

B. *Related Proceedings*

The parties identify five civil actions that have been consolidated into one, *Webroot, Inc. v. AO Kaspersky Lab*, No. 6:22-cv-00243-ADA-DTG (W.D. Tex.), which is pending. *See* Pet. 3; Paper 5, 1.

C. *The ’844 Patent*

The ’844 patent is titled “Automatic Threat Detection of Executable Files Based on Static Data Analysis” and describes methods in which an executable file may be analyzed without being decrypted or unpacked. *See* Ex. 1001, 1:20–34. This is done by extracting “static data points” from the file and applying to them a classifier that was trained from known malicious, benign, and unwanted executable files. *See id.*

The general architecture of the system is shown in Figure 1, reproduced below:

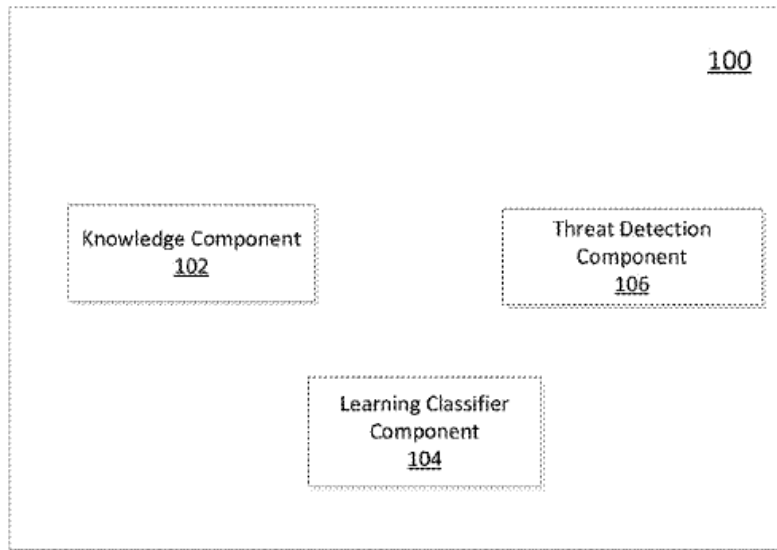


FIG. 1 illustrates “an exemplary system 100 showing [an] interaction of components for implementation [of] threat detection.” Ex. 1001, 1:48–50.

The “system 100 comprises a knowledge component 102, a learning classifier component 104, and a threat determination component 106.” Ex. 1001, 3:23–25.

“To classify executable files . . . the knowledge component 102 collects static data on a large variety of executable files.” Ex. 1001, 4:22–24. The data collected “yields a knowledgebase that may be used to periodically train a classifier, e.g. learning classifier component 104,” which “may be used to classify an executable file as one of the following classifications: malicious, benign or potentially unwanted.” *Id.* at 4:34–39. “Continuous update of the data maintained by the knowledge component 102 in conjunction with on-going re-learning/re-training of the learning classifier component 104 ensures that system 100 is up-to-date on current threats.” *Id.*

at 4:56–60.

The collected data “is analyzed to identify static data points that may indicate one of a malicious file, a benign file or a potentially unwanted file.” Ex. 1001, 5:1–4. 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.” *Id.* at 5:64–6:1. Table 1.1 of the ’844 patent, showing possible static data points, 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	language	original file	Entry Point Anomaly
version			
file version	file flags masks	private build	certificate Validity
number			
product version	file flags	product name	Certificate Exception
number			
size of heap	file OS	special build	Code Characteristics Anomaly
size of stack	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	Number of Functions
		Subject	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	PE Magic Validity
		section strings	
	resource code		Resource Exception
	page		
	resource size		VR Code Ratio Anomaly
	DLL		Import Exception
	characteristics		

Table 1.1 shows possible static data points, organized by category.

The learning classifier component 104 “interfaces with the knowledge component 102 and a threat detection component 106 for the system 100 to evaluate an executable file as a possible threat,” such as by applying “programming operations or machine learning processing to evaluate static data points extracted from a file to be analyzed.” Ex. 1001, 7:43–48. It “builds a learning classifier based on information from the knowledge component 102 including the extracted static data points of a file and the information used to train/re-train the learning classifier.” *Id.* at 7:49–53.

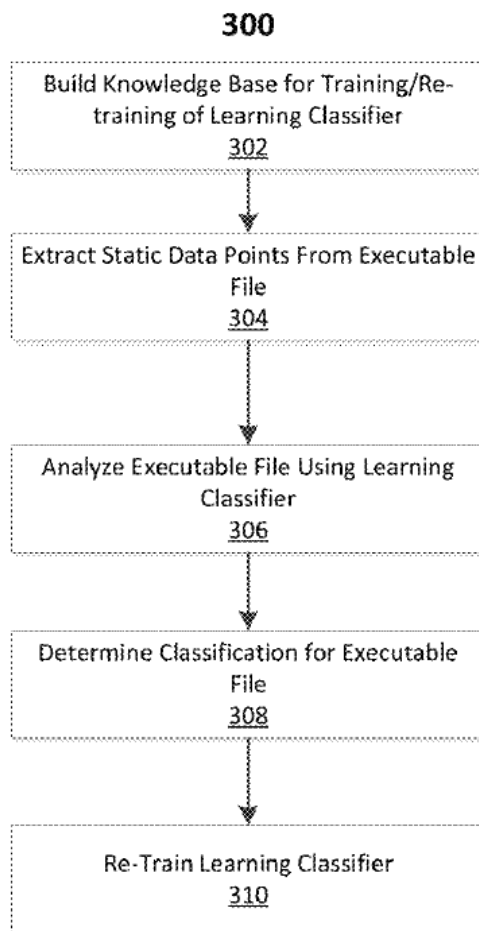
“[T]he learning classifier can adaptively set features of a learning classifier based on the static data points extracted for a file.” Ex. 1001, 7:57–59. For example, “ranges of data can be identified for static data points that may enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier.” *Id.* at 7:59–62. As a more specific example, “where a static data point extracted for evaluation is a file size . . . , the learning classifier may turn on/off features for evaluation based on whether . . . the file size . . . is within a certain range.” *Id.* at 7:62–66.

“The learning classifier component 104 processes and encodes collected/extracted data from a file to make the static data points suitable for processing operations” and “the different encoded data may be combined into one or more final feature vectors.” Ex. 1001, 8:22–24, 8:45–46.

The learning classifier component 104 then “provides the one or more feature vectors as input to a support vector machine (SVM)” which “may then perform data analysis and pattern recognition on the one or more feature vectors” and “build a probabilistic model that indicates whether or not a file may be malicious.” Ex. 1001, 9:41–48.

The threat detection component 106 “evaluates results of the processing performed by the learning classifier component 104 to make a determination as to whether the executable file is malicious, benign or a potentially unwanted file/application.” Ex. 1001, 10:1–5. For example, “a final score (e.g., probability value/evaluation) [may be] determined based on feature vector processing performed by the learning classifier component 104 and compared with a probability threshold value for determining whether an executable file is malicious.” *Id.* at 10:16–21.

Figure 3 is a flowchart showing an exemplary threat detection method 300:



“FIG. 3 illustrates an exemplary method 300 for implementation of threat detection systems and methods.” Ex. 1001, 1:54–55.

The method begins at “operation 302 where a knowledge base is built for training/retraining of a learning classifier used to detect threats in executable files . . . from data collected and evaluated related to known malicious executable files, known benign executable files and potentially unwanted executable files” by knowledge component 102. Ex. 1001, 11:53–55. The knowledge base “may be used to automatically train/re-train one or more learning classifiers used to evaluate threats in executable files based on the known malicious executable files, known benign executable files and the potentially unwanted executable files.” *Id.* at 11:60–64.

“When an executable file is identified for evaluation,” the method proceeds to operation 304, “where a one or more static data points are extracted from [the] executable file” and the file “is analyzed using machine learning processing as described with respect to the knowledge component 102 . . . to collect/extract static data points . . . for evaluation.” Ex. 1001, 12:4–10. This may happen with or without decrypting or unpacking the file. *Id.* at 12:10–12.

At operation 306, “the executable file is analyzed for threats” by “applying a learning classifier (e.g., the learning classifier generated during performance of operation 302) to the plurality of static data points extracted from the file.” Ex. 1001, 12:36–44. The system may “generat[e] at least one feature vector from the plurality of static data points extracted using the learning classifier trained by the knowledge base” and the “data may be parsed and encoded for machine learning processing.” *Id.* at 12:48–53.

As noted above, “generation of the feature vector may comprise selectively setting features of the learning classifier based on the one or more of [the] static data points extracted” and “[f]eatures of the generated feature

vector may be weighted based on classified categories identified by the knowledge base . . . and the plurality of static data points extracted from the file.” Ex. 1001, 12:54–60. In addition, “one or more features of the feature vector may be selectively turned on or off based on evaluation of whether a value of a static data point is within a predetermined range.” *Id.* at 12:60–63. The feature vector is evaluated “using linear or nonlinear support vector processing to determine a classification for the executable file, for example whether the executable file is harmful, benign, or unwanted.” *Id.* at 13:1–4.

At operation 308, “a determination is made as to a classification of the executable file,” i.e., “whether the executable file is harmful (e.g., malicious, malware) or not based on results of the analysis of the executable file (e.g., using machine learning processing by a learning classifier).” Ex. 1001, 13:5–11. The “results of the analysis of [the] executable file may be data obtained from learning classifier, such as, for example an SVM, processing data.” *Id.* at 13:11–13. The classification “may be based on any type of analytical, statistical or graphical analysis, or machine learning processing.” *Id.* at 13:27–29.

“At any point in time, operation 310 may occur where a learning classifier used for threat detection processing is re-trained” to “ensure that the threat detection application/service is up to date and able to accurately detect new threats.” Ex. 1001, 13:34–36.

Claims 1, 10, and 17 are independent. Claim 1 is directed to a computer-implemented method, claim 10 is directed to a “computer-readable storage device containing instructions, that when executed on at least one processor, caus[e] the processor to execute a process” corresponding to the

method of claim 1, and claim 17 is a corresponding system claim. Claim 1 is thus representative:

1. A computer-implemented method comprising:

extracting a plurality of static data points from an executable file without decrypting or unpacking the executable file, wherein the plurality of static data points represent predefined character strings in the executable file;

generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files, known benign executable files, and known unwanted executable files, wherein the collection of data comprises at least a portion of the plurality of static data points, and wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range; and

evaluating the feature vector using support vector processing to determine whether the executable file is harmful.

We understand “generating a feature vector from the plurality of static data points using a classifier . . . wherein one or more features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range” to correspond to the Specification’s description of “adaptively set[ting] features of a learning classifier based on the static data points extracted for a file.” Ex. 1001, 7:57–59.

D. Asserted Grounds of Unpatentability

Petitioner’s challenge raises the following grounds:

Claims Challenged	Basis	References
1–20	§ 103	Ideses ¹
1, 8–10, 17	§ 103	Mori ²
1–20	§ 103	Ideses and Mori

Petitioner also relies on the Declaration of Dr. Gene Tsudik, filed as Exhibit 1004. Patent Owner relies on the Declaration of Sam Malek, Ph.D., filed as Exhibit 2001.

II. ANALYSIS

A. Level of Ordinary Skill in the Art

Petitioner contends that one of ordinary skill in the art at the time of the alleged invention “would have a Bachelor’s degree in Computer Science, Computer Engineering, Electrical Engineering, or an equivalent discipline, and at least one year’s worth of experience relating to static data analysis and malware detection,” or “[i]n the alternative, . . . two or more years of work experience relating to static data analysis and malware detection.” Pet. 11 (citing Ex. 1004 ¶ 16). Patent Owner “does not dispute Petitioner’s articulation of the level of ordinary skill.” Prelim. Resp. 30.

For purposes of this decision, we adopt Petitioner’s proposal, which we find generally consistent with the disclosures of the ’844 patent.

¹ U.S. Patent Appl. Pub. No. 2015/0213376 A1 (Exhibit 1005).

² Japanese Patent Appl. Pub. No. P2012-27710A (Exhibit 1006).

B. Claim Construction

We construe claims using the standard that would be applied in a civil action under 35 U.S.C. § 282(b), giving terms their ordinary and customary meaning to one of ordinary skill in the art in view of the specification and prosecution history. *See* 37 C.F.R. § 42.100(b).

Petitioner asserts that the Petition is based on “constructions that Patent Owner has advocated in the Litigation,” and that it thus “does not present any controversy over claim meaning.” Pet. 8. For its part, Patent Owner “agree[s] that no claim terms need be construed for this petition,” and that “all claim terms should carry their plain and ordinary meaning in light of the intrinsic record.” Prelim. Resp. 28.

We find that no express claim construction is necessary to resolve the institution question. *See Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F.3d 1013, 1017 (Fed. Cir. 2017) (explaining that construction is needed only for terms that are in dispute, and only as necessary to resolve the controversy).

C. Patentability

A claim is unpatentable under 35 U.S.C. § 103 if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious to a person having ordinary skill in the art to which said subject matter pertains. *See KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations, including (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of skill in the art; and (4) where in evidence, so-called secondary considerations, including commercial

success, long-felt but unsolved needs, failure of others, and unexpected results. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

1. *Ideses*

Ideses “relates to . . . methods and systems for generating classifiers for software applications and, more specifically . . . to methods and systems for generating classifiers for software applications based on large feature vectors.” Ex. 1005 ¶ 2.

Petitioner refers to the method Ideses depicts in Figure 1, which is reproduced below.

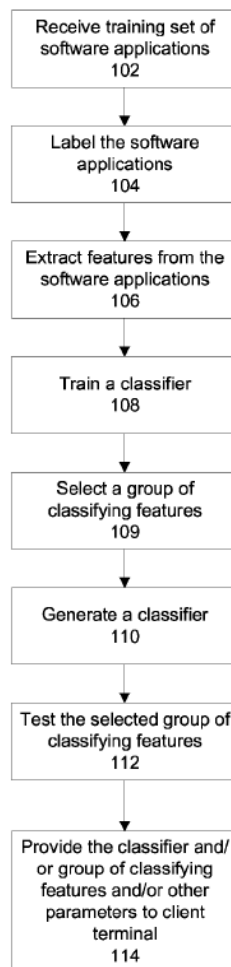


Figure 1 is a “flowchart of a method of generating a pruned feature list for classification.” Ex. 1005 ¶ 39.

At step 102, the system receives multiple software applications for training a classifier and, at step 104, the applications may be “labeled with a classification type based on desired software application classification categories.” *See* Ex. 1005 ¶¶ 80–81.

At step 106, features may be extracted from the software applications by a feature extractor module and “[o]ptionally, a feature vector [may be] extracted.” *See* Ex. 1005 ¶ 86. The extracted features may be any feature that describes the software application, including static meta data, information regarding the executable code, or behavioral features. *See* Ex. 1005 ¶ 89. The features may be stored as feature vectors, which may be used for training a classifier. *See id.* ¶ 91.

At step 108, “a classifier [may be] trained based on the set of extracted features (block 106) and software classification labeling (block 104).” Ex. 1005 ¶ 94. At step 109, “a group of classifying features is selected, for example, by a combination and/or selection module 306.” *See id.* ¶ 98. “[E]xtracted features [may be] identified for pruning to generate the group of classifying features,” and “the selection [may be] based on identifying a classification effectiveness rank for each of the features of the training software applications” or “based on identifying resource requirements for each of the features of the training software applications.” *See id.* ¶¶ 98, 99.

The selection module may use “the parameters and/or coefficients gathered during the classification training” and “the group of classifying features [may be] selected based on the significance of the extracted features,” such that “[s]ignificant features, such as those that have a large effect on the classification outcomes, may be retained” and “[i]nsignificant

features, such as those that have a small, negligible or no effect on classification outcome may be pruned.” Ex. 1005 ¶¶ 101, 106.

“[F]eatures (or their equivalents) that do not contribute (or do not significantly contribute) to the classification process [may be] pruned or not included,” “features that correspond to coefficients having a value of zero [may be] pruned or not included, for example, when classification is performed based on a support vector machine (SVM),” and “features with a low magnitude value of the coefficient [may be] identified for pruning and/or are not included.” Ex. 1005 ¶ 107.

At step 110, “a classifier is generated, for example, by a classifier generating module” that may be “based on the group of classifying features.” Ex. 1005 ¶¶ 115–116. The feature set used to generate the trained classifier may be pruned. *See id.* ¶¶ 116–117.

Step 112 is an optional step of evaluating the “classifier performance based on the pruned feature set [as] compared to classifier performance based on the complete feature set” and “[t]he pruning may be selectively adjusted based on the testing.” *See* Ex. 1005 ¶¶ 118–119.

Finally, at step 114, “the group of classifying features, the pruned feature vector, selected coefficients, and/or trained classifier are provided to [a] mobile device” “for run-time classification of software applications by the mobile device.” *See* Ex. 1005 ¶¶ 120–122.

2. *Mori*

Mori relates to “a software detection method, apparatus and program for detecting malware or communications transmitting and/or receiving malware using supervised machine learning systems.” Ex. 1006 ¶ 1. The method “focuses on readable strings contained in software files, collects information (characteristic vectors) that characterizes malware, such as

messages contained in programs that constitute malware, host names, execution command names, function names and library names used for communication,” and “compares this content with strings contained in files being inspected to detect malware, allowing for the timely detection of malware and communications that include malware at a high rate.” *Id.* ¶ 23.

Mori describes the process as follows: “[f]irst, the file or communication data to be sent or received,” called the “Object,” is “recorded by the Object recording unit”; next, “a word extraction unit . . . extracts the printed and readable byte strings as words from the Object . . . and uses it as a word set”; then, “by configuring a feature vector for each Object based on the . . . word set, and applying a machine learning system to the . . . feature vector, the determination function determines whether the Object is malware based on newly observed feature vectors.” Ex. 1006 ¶¶ 27–30. “[T]he feature vector can optionally be defined as whether a word existed or not by assigning it ‘0,’ or ‘1,’ the frequency per word, the normalized frequency per word, or the tf-idf value per word.” *Id.* ¶ 30.

Mori’s system is depicted in its Figure 1, reproduced below:

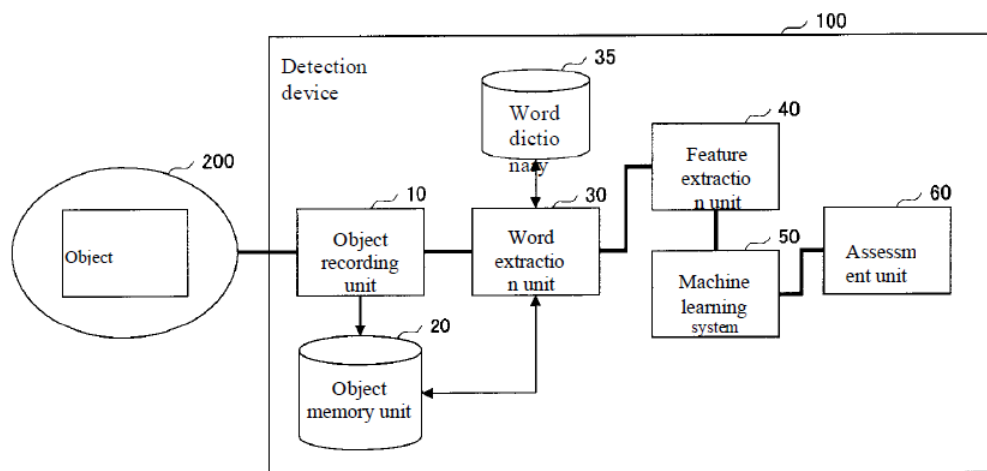


Figure 1 is an “[e]xample of a system configuration.” Ex. 1006, 25.

As shown above, “[d]etection device 100 is connected to network 200 and consists of Object recording unit 10, Object storage unit 20, word extraction unit 30, word dictionary 35, feature extract unit 40, machine learning system 50, and assessment unit 60.” Ex. 1006 ¶ 37.

“Word extraction unit 30 retrieves the Object from Object storage unit 20, analyzes it, and extracts a string of consecutive characters with an initial printable character code of X characters or more,” where “printable” refers, for example, “to the character codes of blank and graphic characters from 0×20 to 0×7e, excluding control characters, in ASCII codes.” Ex. 1006 ¶ 39. The strings are processed to produce word candidates such as “text,” “suvn,” and “set.” *See id.* ¶¶ 41–49. Then, “[w]ord extract unit 30 uses word dictionary 35 prepared in advance for the word candidates . . . , extracting only those words that match the words in word dictionary 35.” *Id.* ¶ 50. This would remove “suvn,” for example.

Next, “[f]eature extraction unit 40 reads out the word set . . . for each file from the memory of word extract unit 30 and configures the feature vector.” Ex. 1006 ¶ 52. For each file X_i ($i=1, 2, \dots, N$), the feature vector is $V_i = \{v_{i1}, v_{i2}, \dots, v_{ij}, \dots, v_{iM}\}$, where, for each word w_j ($j=1, 2, \dots, M$), $v_{ij} = g(w_j, X_i, X, a)$. Ex. 1006 ¶ 52. “X” is “the set of all files” and “a” is “an optionally definable weight parameter.” *Id.* As noted above, “the feature vector can optionally be defined as whether a word existed or not by assigning it “0,” or “1,” the frequency per word, the normalized frequency per word, or the tf-idf value per word.” *Id.* ¶ 30.

Finally, “[a]ssessment unit 60 applies machine learning system 50 to the feature vector V_i ” to classify the file as “malware” or a “normal file.” Ex. 1006 ¶¶ 54–55. “Specifically, each ‘malware’ and ‘normal file’ file

feature vector is learned in advance, and the newly obtained file is determined to be ‘malware’ or ‘normal file’ based on that feature vector.”

Id. ¶ 55. Mori explains that “supervised learning” is “a highly accurate method” for estimating the class for a newly observed vector, “with a variety of known methods widely used, including support vector machines (SVMs), naïve Bayesian methods, neural networks, and decision trees.” *Id.* ¶ 57.

3. *Analysis*

The only argument Patent Owner makes on the merits is that Ideses and Mori do not render the claims obvious because they fail to teach or suggest turning features on or off at the classification stage based on the executable being classified. *See* Prelim. Resp. 34–38 (Ideses), 41–47 (Mori).

Specifically, Patent Owner argues that the claims require “extracting . . . static data points from an executable,” “generating a feature vector from the . . . static data points **using a classifier trained to classify (past tense)** the . . . static data points,” “wherein one or more features of the feature vector are . . . turned on or off based on . . . [the] extracted static data points,” and “evaluating the feature vector . . . to determine whether the executable . . . is harmful” (i.e., classifying the executable). Prelim. Resp. 35 (alterations in original). According to Patent Owner, “[b]ecause the ‘executable’ for which the feature vector is generated is the executable being classified at the classification stage, the requirement of turning features of that feature vector ‘on or off based on . . . [the] extracted static data points’ from that executable must necessarily happen at the classification stage too.” *Id.* (alterations in original).

Patent Owner asserts that “[t]he benefit of turning features on or off at the classification stage is that classifier performance can be optimized based on the executable being classified” and “the classifier is not locked into

evaluating the full set of features that was predetermined at the training stage without regard to the executable being classified at the classification stage.”

Prelim. Resp. 36 (emphasis omitted).

Patent Owner further argues that, unlike the claimed method, “Ideses teaches feature selection at the training stage before the executable being classified is ever seen” and Mori “determines a set of features at the training stage . . . and trains a classifier to classify executables based on that set of features.” Prelim. Resp. 36, 43 (emphasis omitted).

We agree with Patent Owner that the claims are directed to the classification stage, as they require “extracting a plurality of static data points from an executable file” and “generating a feature vector from [that] plurality of static data points *using a classifier*” (emphasis added). The claims also recite that the feature vector generated from the static data points that were extracted from the executable file is used “to determine whether the executable file is harmful,” confirming that the claimed method is about classification of an executable file, not creation of a classifier.

We also agree with Patent Owner that neither Ideses nor Mori teaches or suggests a classification method in which one or more features of a feature vector generated from a plurality of static data points extracted from a file being analyzed are selectively turned on or off based on the value of the static data point.

As explained above, Ideses describes “systems for generating classifiers” (Ex. 1005 ¶ 2) in which “extracted features [may be] identified for pruning to generate the group of classifying features,” and “the selection [may be] based on identifying a classification effectiveness rank for each of the features of the training software applications” or “based on identifying

resource requirements for each of the features of the training software applications.” *See id.* ¶¶ 98, 99. “[T]he group of classifying features [may be] selected based on the significance of the extracted features,” such that “[s]ignificant features, such as those that have a large effect on the classification outcomes, may be retained” and “[i]nsignificant features, such as those that have a small, negligible or no effect on classification outcome may be pruned.” Ex. 1005 ¶¶ 101, 106.

But the selection and pruning described in Ideses concerns the creation of the classifier, where the features that are selected or pruned are either included or omitted from the classifier. Petitioner fails to show how this teaches or suggests “generating a feature vector from [a] plurality of static data points” that were “extract[ed] . . . from an executable file” and “using a classifier trained to classify the plurality of static data points based on a collection of data comprising known malicious executable files” where features are “selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range.” Even if Ideses’ omission of insignificant features could be “turn[ing] [them] on or off,” that happens in the creation of the classifier, not “based on . . . a value of one or more static data points from the plurality of extracted static data points” for the purpose of “determin[ing] whether the executable file is harmful,” as recited in each challenged claim.³

³ Ideses also describes using its classifier to perform classification (*see* Ex. 1005 ¶¶ 123–147), but Petitioner does not rely on that portion of the reference, and it does not appear to describe turning features on or off based on the value of the extracted data points.

Mori describes a process that includes use of a classifier to detect malware, in which a feature vector is derived from the object being analyzed. However, we conclude that Petitioner has not shown how Mori teaches or suggests that “features of the feature vector are selectively turned on or off based on whether a value of one or more static data points from the plurality of extracted static data points is within a predetermined range.”

As described above, Mori discloses a feature extraction unit that locates words in an object to be analyzed and produces a feature vector that includes each word and can “optionally be defined as whether a word existed or not by assigning it ‘0,’ or ‘1,’ the frequency per word, the normalized frequency per word, or the tf-idf value per word.” Ex. 1006 ¶ 30.

Petitioner argues that Mori’s explanation “that the feature vector can be defined ‘as whether a word existed or not by assigning it ‘0’ or ‘1’” means that “features are selectively turned on or off based on whether or not a word in the predetermined dictionary is present in the Object or not.” Pet. 54. We do not agree, because we see nothing in Mori to suggest that a feature in the vector assigned a value of 0 is not used in the evaluation. Petitioner does not adequately explain how assigning a feature a value of 0 is the same as turning it “off.” Petitioner also does not adequately explain how Mori assigning a value of 0 or 1 based on “whether a word existed or not” would amount to making a determination based on whether “a value” of the data point “is within a predetermined range,” as required by the claims. For example, it is unclear how a word that does not exist in the Object might have a “value.”

Petitioner also argues that Mori describes how “words that are present in each file are represented as ‘{word number: tf-idf value}’ in the feature vector” and that “if a word is not present, the ‘static data point’ for that word

is ‘turned off’ since it is not included in the generated feature vector for that file (i.e., the tf-idf value is 0 because the tf portion is 0).” Pet. 56. We also disagree with that analysis. The claims require that the feature vector be generated from the plurality of static data points extracted from the executable file, so, if the word “is not present” in the executable file, it would not appear to be a “static data point” that could be extracted from the executable file or from which the feature vector would be generated. As quoted by Petitioner, Mori explains that “*words that are present in each file* are represented . . . in the feature vector,” which we understand to also mean that words *not* present in the file are *not* represented in the feature vector. Pet. 56 (emphasis added). Thus, for words that are not present in the Object, there does not appear to be a “feature[] of the feature vector [to be] selectively turned on or off.” And, as above, Petitioner also does not explain how, under this theory, “features of the feature vector [would be] selectively turned on or off” based on whether “a value” of the data point “is within a predetermined range.” It is unclear how a word that “is not present” would have “a value.”

For these reasons, we find that Petitioner has failed to show that Ideses or Mori teaches or suggests “extracting a plurality of static data points from an executable file” and “generating a feature vector from the plurality of static data points using a classifier trained to classify the plurality of static data points based on a collection of data,” where “one or more features of the feature vector are selectively turned on or off based on whether a value of

one or more static data points from the plurality of extracted static data points is within a predetermined range,” as recited in each independent claim.⁴

III. CONCLUSION

We determine that Petitioner has not established a reasonable likelihood of proving that at least one challenged claim of the ’844 patent is unpatentable over the cited prior art. We accordingly do not institute an *inter partes* review.⁵

IV. ORDER

For the reasons given, it is ORDERED that no *inter partes* review is instituted in this proceeding.

⁴ Claim 17 recites “character strings,” instead of “static data points,” but Petitioner does not argue that the variation makes a difference in the analysis (*see, e.g.*, Pet. 46 (regarding element [17.a]: “*See* element [1.a], above.”), so we consider “character strings” and “static data points” interchangeable for purposes of this decision.

⁵ Because we deny the Petition on the merits, we do not reach Patent Owner’s request that we exercise our discretion to deny the Petition based on the state of the related litigation. *See* Prelim. Resp. 3–21. For the same reason, we deny Petitioner’s late request (*see* Ex. 3001) to submit a new stipulation.

IPR2023-00528
Patent 10,599,844 B2

For PETITIONER:

Benjamin Hershkowitz
Ryan Iwahashi
GIBSON, DUNN & CRUTCHER LLP
bhershkowitz@gibsondunn.com
riwahashi@gibsondunn.com

For PATENT OWNER:

Brian Eutermoser
Russell Blythe
Mikaela Stone
KING & SPALDING LLP
beutermoser@kslaw.com
rblythe@kslaw.com
mikaela.stone@kslaw.com