

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

TREND MICRO, INC.,
Petitioner

v.

WEBROOT INC.,
Patent Owner.

IPR2023-00662
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*

Trend Micro Inc. (“Petitioner”) filed a Petition requesting *inter partes* review of claims 1–20 of U.S. Patent No. 10,599,844 B2 (Ex. 1001, “the ’844 patent”). Paper 2 (“Pet.”). Open Text Inc. (“Patent Owner”) filed a Preliminary Response. Paper 6 (“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 the following district court cases as related to this proceeding:

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. CrowdStrike, Inc. et al.,
Case No. 6:22-cv-00241 (WDTX); and

Webroot, Inc. et al. v. AO Kaspersky Lab,
Case No. 6:22-cv-00243 (WDTX); and

Webroot, Inc. v. Forcepoint LLC,
Case No. 6:22-cv-00342 (W.D. Tex.).

See Pet. 1–2; Paper 4, 1.

We previously denied institution of an *inter partes* review of the '844 patent in IPR2023-00528.

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:

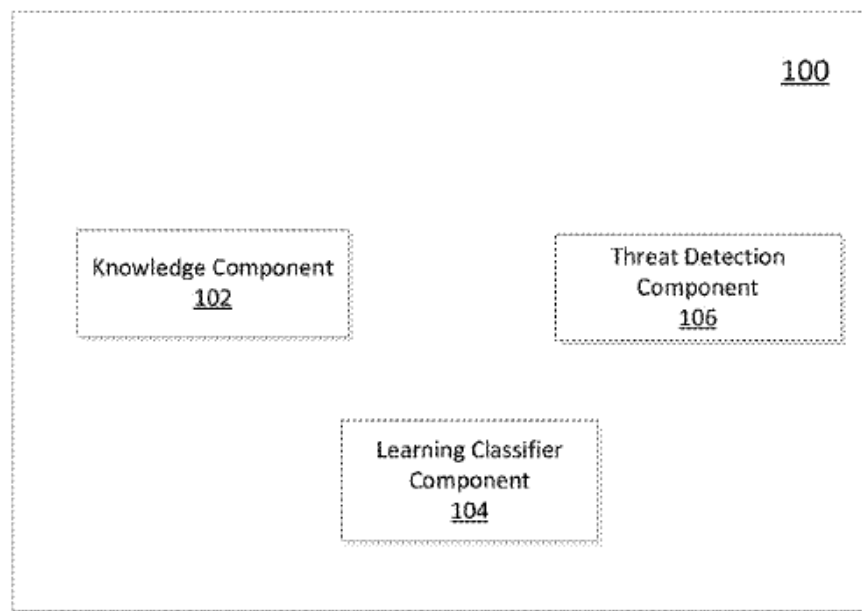


FIG. 1 illustrates “[an] interaction of components for implementation [of] threat detection.” Ex. 1001, 1:48–50.

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–

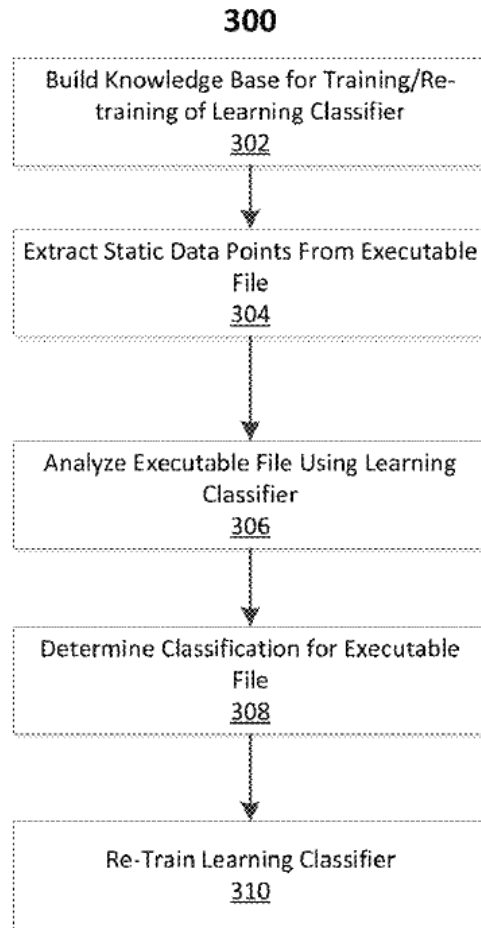
59. For example, “ranges of data can be identified for static data points that may enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier.” *Id.* at 7:59–62. As a more specific example, “where a static data point extracted for evaluation 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, reproduced below, 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 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 then 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 35 U.S.C. §	Reference(s)
1–5, 8, 10–14, 17, 19	103	Tahan, ¹ Menahem ²
6–7, 9, 15–16, 18, 20	103	Tahan, Menahem, Schultz ³

Petitioner also relies on the Declaration of Seth James Nielson, filed as Exhibit 1002. 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 had a bachelor’s degree in computer science, or a related field, or an equivalent technical degree or equivalent work experience, and an additional one to two years of education or experience in the field of computer security topics such as malware/exploit detection and/or machine learning” and that “[m]ore education can supplement practical experience and vice versa” Pet. 9 (citing Ex. 1002 ¶ 94). Patent Owner “does not dispute Petitioner’s articulation of the level of ordinary skill.” Prelim. Resp. 29.

¹ Gil Tahan, et al., *Mal-ID: Automatic Malware Detection Using Common Segment Analysis and Meta-Features*, Journal of Machine Learning Research 13, 949–979 (April 13, 2012) (Exhibit 1003).

² Eitan Menahem, et al., *Improving Malware Detection by Applying Multi-Inducer Ensemble*, Computational Statistics & Data Analysis 53, 1483–1494 (2009) (Exhibit 1004).

³ U.S. Patent App. Pub. No. 2009/0254992 A1 (Exhibit 1005).

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

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 “all claim terms should be given their plain and ordinary meanings.” Pet. 9. 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. 27.

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

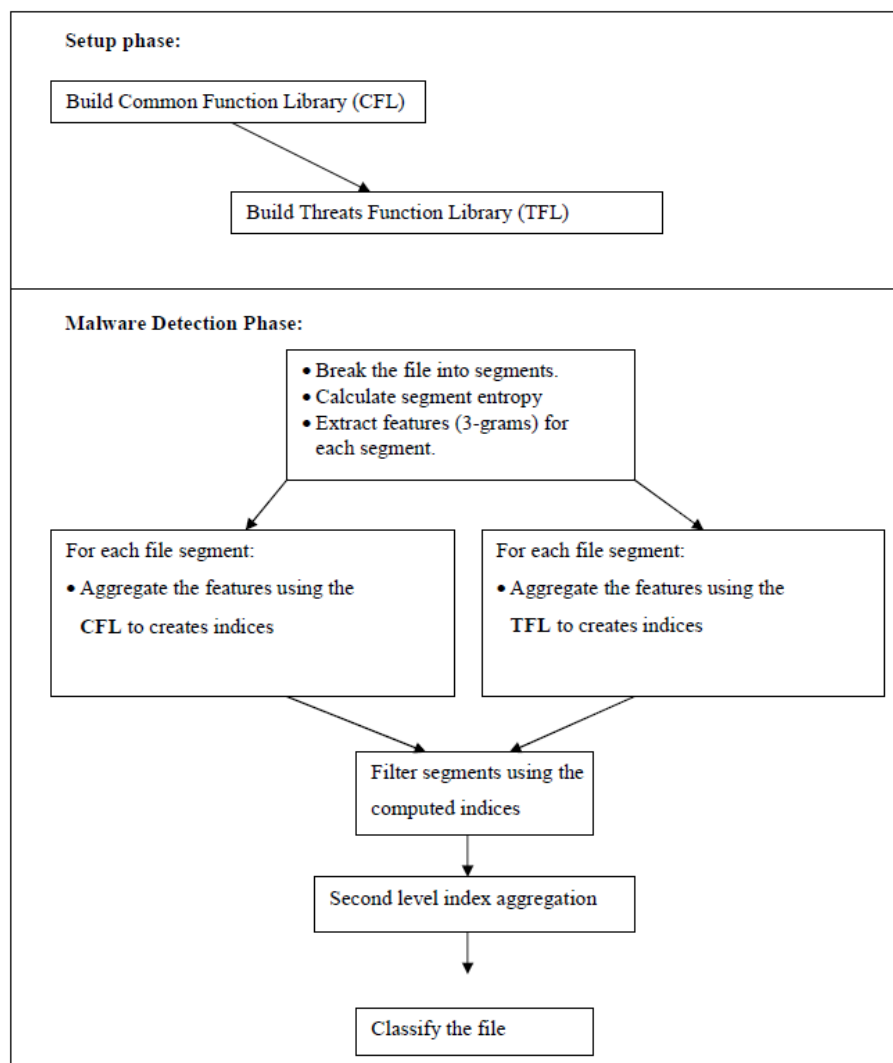
A claim is unpatentable under 35 U.S.C. § 103 if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious to a person having ordinary skill in the art to which said subject matter pertains. *See KSR Int’l Co. v. Teleflex 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. *Summary of Tahan*

Tahan is an academic paper that “proposes several novel methods, based on machine learning, to detect malware in executable files without any need for preprocessing, such as unpacking or disassembling.” Ex. 1003, 1.

The primary method is illustrated in Tahan’s Figure 1, shown below:



“Figure 1: The Mal-ID method for detecting new malware applications.” Ex. 1003, 8.

As depicted, the Mal-ID method starts with a Setup Phase that establishes the Build Common Function Library (“CFL”) and Build Threats Function Library (“TFL”). It then moves to a Detection Phase, in which short segments comprising a number of “3-grams” are extracted from the file being analyzed. *See* Ex. 1003, 9. The method then derives three “meta-features” for each segment: Spread, which “approximates the maximum number of occurrences of a segment within different relative locations in train sets”; MFG, which is “the maximum total number of file-groups that contain the segment”; and Entropy, which is “the entropy measure of the bytes within a specific segment candidate.” *Id.*

Next, some segments are discarded based on the Entropy being too high or low, segments for which a CFL-MFG index is larger than zero are discarded (because they may appear in benign files), and segments for which a TFL-MFG index is smaller or equal to the ThreatThreshold are discarded (removing segments that do not appear in malware). *See id.* at 10. In addition, segments for which a TFL-Spread index is zero or larger than a threshold are also discarded (eliminating segments likely to belong to a distinct library function). *Id.*

The narrowed set of segments is then used to classify the file: “[i]f there are at least X segments found in the [TFL] and not in the CFL then the file is malware; otherwise . . . the file [is considered to be] benign.” *Id.* at 11.

2. *Menahem*

Menahem is an academic paper that surveys methods for detecting malware using machine learning methods and “suggest[s] using an ensemble method that will combine the results of the individual classifiers into one final result to achieve overall higher detection accuracy.” Ex. 1004, 1.

3. *Analysis*

Generally speaking, Petitioner argues that Tahan discloses the elements of the independent claims, but also cites Menahem for the concept of extracting static data points that represent predefined character strings. *See, e.g.*, Pet. 13–23. We will focus on the issues raised by Patent Owner, as well as several of our own.

a. *Training vs. Classification Stages*

Patent Owner first argues that “[b]ecause the step of ‘generating a feature vector’ happens at the classification stage, the requirement that this step include turning features ‘on or off’ must likewise happen at the classification stage” and that “[n]either Tahan nor Menahem (alone or in combination) suggests the idea of turning features on or off at the classification stage” because “each teaches only the traditional idea of performing feature selection at the training stage.” Prelim. Resp. 41–42.

We do not agree with this analysis. As explained above, the Mal-ID method has a Setup Phase that establishes the CFL and TFL libraries. *See* Ex. 1003, 8. Then it shifts to the Detection Phase, in which the file being analyzed is broken into sections and features are extracted. *Id.* The Spread, MFG, and Entropy meta-features are then used to discard segments, leaving a subset of the original segments, and that subset is used to classify the file as benign or malware depending on the number of those segments are found in the TFL and CFL. *See id.* at 9–11.

We conclude that, contrary to Patent Owner’s arguments, Tahan’s method of discarding segments using the TFL and CFL is generating a feature vector (the one used in the classification) from the plurality of static data points (the segments) using a classifier (the TFL/CFL) trained to classify the plurality of static data points, wherein one or more features of the feature

vector are selectively turned on or off (discarded). As in the claims, a feature vector is generated from the file, features (segments) are discarded, and the remaining features (segments) are used to classify the file.

b. Predefined Character Strings

Patent Owner's second argument is that "Tahan does not suggest extracting static data points that represent predefined character strings, and Petitioner fails to explain how Tahan could be modified to use that idea or why a POSITA would have been motivated to make that modification."

Prelim. Resp. 53. Patent Owner argues that the applicant introduced the character strings limitations during prosecution to overcome a rejection based on a reference called Wojnowicz that operated like Tahan, in that it split the file being analyzed into chunks of fixed length and then performed an entropy analysis on those chunks. *See id.* at 53–54 (citing Ex. 1007, 156).

According to Patent Owner, "[t]he applicant explained that Wojnowicz's decomposition of files into 'fixed-length file chunks' does not satisfy the new limitation respecting a 'predefined character strings' because 'fixed-length file chunks do not represent predefined character strings'" but, instead, "are 'random character string[s],' and extracting them 'does not include comparison of the random character string to a predefined character string.'" *Id.* at 54 (citing Ex. 1007, 156). Patent Owner contends that "Petitioner now argues that Tahan satisfies the 'extracting' limitations by performing the same operation as Wojnowicz: splitting the executable into fixed-length chunks." *Id.* (citing Petition 13–16).

We agree with Patent Owner that Tahan's disclosure of extracting "short segments" does not disclose extracting static data points that represent predefined character strings. We see nothing to suggest that the segments

represent predefined strings, and the method operates using the “meta-features” derived from the segments, not predefined character strings. *See* Ex. 1003, 9–10. We have reviewed the Petition and the Nielsen declaration and find that, to the extent there is discussion in Tahan of extracting strings, it exists in the portion of the paper providing a background of machine learning malware detection techniques, not as part of the Mal-ID method Tahan discloses. Petitioner is not free to pick-and-choose aspects of different malware detection methods without clearly explaining how or why they would be combined, which it has not done.

Patent Owner also argues that Petitioner’s reliance in the alternative on Menahem is misplaced because Petitioner does not explain “how Tahan’s Mal-ID system would be modified to use PE headers as the features” or “why a POSITA would have been motivated to make that modification.” Prelim. Resp. 47–58. We agree. As with the argument based on the background portion of Tahan, Petitioner does not explain how or why the various malware detection methods described in Menahem could or would be meshed with Tahan’s Mal-ID method.

c. Value Within a Predetermined Range

Although Patent Owner does not raise the issue, we also find that Petitioner has not shown that Tahan discloses dropping segments “based on whether a value of one or more [segments] . . . is within a predetermined range.” The reference discloses dropping segments based on entropy and the CFL-MFG, TFL-MFG, and TFL-Spread indices, not the values of the segments themselves. Petitioner does not explain how or why the meta-features or the indices should be considered “a value of one or more static data points.”

d. Known Unwanted Executable Files

In addition, and although Patent Owner also does not raise this issue, we find that Petitioner has not shown that Tahan teaches or suggests “classify[ing] 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.”

The ’844 patent explains that “an executable file may be benign in the sense that it is not harmful to system 100 but may also be classified as potentially unwanted as it might be installed without explicit user consent.” Ex. 1001, 4:41–44. This third category thus concerns files that are benign (i.e., not malicious), but just undesired. We find Petitioner’s assertion that “spyware” is an “unwanted” file (*see* Pet. 18–19) to be incorrect, because that type of software would fall into the known malicious category.

Tahan classifies files based on known malicious files (using the MFL) and known benign files (using the CFL), but Petitioner has not identified a disclosure of a category that would correspond to the claimed “known unwanted executable files.”

e. Support Vector Processing

Lastly, we further find, although Patent Owner does not argue it, that Petitioner has not shown Tahan to disclose “evaluating the feature vector using support vector processing to determine whether the executable file is harmful.” As explained above, Tahan classifies a file as malignant or benign based on the number of segments of the reduced set that are found in the TFL and CFL. *See* Ex. 1003, 11 (“Classify—If there are at least X segments found in the malware train set (TFL) and not in the CFL then the file is malware; otherwise consider the file as benign.”).

For support vector processing (“SVP”), Petitioner points to the portion of Tahan that surveys existing machine learning malware techniques and mentions support vector machines (“SVM”). *See* Pet. 22 (citing Ex. 1003, 2–5⁴). But the fact that SVMs were used in existing systems is insufficient to show that it would have been obvious to use a SVM, or SVP, in Tahan’s Mal-ID system, where Mal-ID uses a different method to evaluate the feature vector to determine whether the executable file is harmful.

Petitioner also argues that “Menahem teaches that its ensemble classifier includes SVMs as base classifiers and builds off prior researches that heavily used SVMs as inducers for the meta-feature vectors that are classified and then passed on to further classifiers.” Pet. 22–23 (citing Ex. 1002 ¶¶ 268, 273–278; Ex. 1004, 2–3). This argument suffers from the flaws identified above, as the cited portions of Menahem are, as in Tahan, background describing the state of the art, and Petitioner fails to explain how those disclosures might be combined with a method like Mal-ID.

III. CONCLUSION

For the foregoing reasons, 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

⁴ Petitioner also cites page 10 of Tahan, but we find no mention of either SVP or SVM there, and Petitioner does not explain where it might be disclosed.

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. 3–18.

IV. ORDER

It is ORDERED that no *inter partes* review is instituted in this proceeding.

For PETITIONER:

Robert M. Hansen
Scott B. Amankwatia
Matthew J. Anderson
THE MARBURY LAW GROUP, PLLC
rhansen@marburylaw.com
samankwatia@marburylaw.com
manderson@marburylaw.com

For PATENT OWNER:

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