

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

Trend Micro, Inc.,

Petitioner

v.

OPEN TEXT INC.

Patent Owner

Case No. IPR2023-01380

Patent No. 11,409,869

**DECLARATION OF SAM MALEK, Ph.D.
IN SUPPORT OF PATENT OWNER'S PRELIMINARY RESPONSE**

TABLE OF CONTENTS

I.	QUALIFICATIONS.....	2
II.	UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES	6
III.	STATE OF THE PRIOR ART	8
IV.	The '869 Patent.....	11
	A. Summary Of The '869 Patent.....	11
	B. Claim Construction	15
	C. Level Of Ordinary Skill In The Art	16
V.	OVERVIEW OF PRIOR ART	17
	A. Overview Of Tahan	17
	1. Background Disclosure	18
	2. Mal-ID Basic	19
	3. Mal-ID With ML	23
	B. Overview Of Menahem	26
	C. Overview Of Forman	27
VI.	THE CITED ART DID NOT DISCLOSE OR RENDER OBVIOUS THE CHALLENGED CLAIMS.....	29
	A. Ground 1: The combination of Tahan, Menahem, and Forman did not render claims 1-4, 8, 10-25, 29, 32-39, 43, 46-53, and 57 obvious.	30
	1. The proposed combination fails to teach turning features on or off at the classification stage	30
	2. The proposed combination fails to teach turning features on or off based on the values of static data points extracted from the executable being classified.	37

3.	The proposed combination fails to teach evaluation of the recited feature vectors using a “support vector machine” or any other “machine learning model.”	42
B.	Ground 2: The combination of Tahan, Menahem, and Schultz did not render dependent claims 5-7, 9, 26-28, 30-31, 40-42, 44-45, 54-56, and 58-59 obvious.	47
VII.	CLAIMS AND/OR LIMITATIONS NOT ADDRESSED IN THIS DECLARATION	47
VIII.	SECONDARY OBJECTIVE INDICIA OF NONOBVIOUSNESS	47
IX.	CONCLUSION	48

EXHIBIT LIST

Exhibit	Description
2001	Declaration Of Sam Malek, Ph.D.
2002	Curriculum Vitae of Sam Malek, Ph.D.
2003	Webroot, Inc., Open Text Inc., v. AO Kaspersky Lab et al., Case No. 6:22-cv-00243-ADA-DTG, Order Granting Amended Scheduling Order, Dkt. 521 (W.D. Tex., Dec. 7, 2023)
2004	Sonrai Memory Ltd. v. Kingston Tech. Co. and Kingston Tech. Corp., Case No. 6:21-cv-1284-ADA, Order Denying Kingston Tech. Co. and Kingston Tech. Corp.’s Opposed Motion to Stay Pending Resolution of Inter Partes Review, Dkt. 94 (W.D. Tex., Oct. 18, 2022)
2005	mCom IP, LLC v. Cisco Systems, Inc., Case No. 6:22-cv-00261-ADA, Order Denying Defendant’s Motion to Stay Pending Inter Partes Review, Dkt. 42 (W.D. Tex., Oct. 20, 2022)
2006	Intellectual Ventures I LLC et al. v. Hewlett Packard Enterprise Co., No. 6:21-cv-00596-ADA, Order Denying Hewlett Packard Enterprise Company’s Opposed Motion to Stay Pending Resolution of Inter Partes Review, Dkt. 104 (W.D. Tex., Dec. 22, 2022)
2007	Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al., Case No. 6:22-cv-243-ADA-DTG, Claim Construction Order, Dkt. 236 (W.D. Tex. Mar. 16, 2023)
2008	Dani Kass, “Catching Up On Patent Litigation With Judge Albright” (Law360 Mar. 14, 2023); https://www.law360.com/pulse/articles/1582438/print?section=pulse/courts
2009	Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al., Case No. 6:22-cv-243-ADA-DTG, Claim Construction Order, Dkt. 391 (W.D. Tex. July 25, 2023)

Exhibit	Description
2010	Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al., Case No. 6:22-cv-243-ADA-DTG, Dec. 9, 2022 Transcript of Motions Hearing, Dkt. 133 (W.D. Tex. Dec. 15, 2022) (excerpted)
2011	Webroot, Inc. and Open Text Inc., v. Trend Micro, Inc., Case No. 6:22-cv-00239-ADA, Complaint for Patent Infringement, Dkt. 1 (W.D. Tex., Mar. 4, 2022)
2012	Webroot, Inc. and Open Text Inc., v. Ao Kaspersky Lab, Case No. 6:22-cv-00243-ADA-DTG, Defendant's Preliminary Invalidity Contentions Appendix C (W.D. Tex., Sept. 14, 2022)
2013	Webroot, Inc. and Open Text Inc., v. Sophos Ltd., Case No. 6:22-cv-00240-ADA-DTG, Defendant Sophos Ltd.'s Final Invalidity Contentions (W.D. Tex., Nov. 1, 2023) (excerpted)
2014	Webroot, Inc. and Open Text Inc., v. Trend Micro Inc., Case No. 6:22-cv-00239-ADA-DTG, Defendant's Final Invalidity Contentions (W.D. Tex., Nov. 1, 2023) (excerpted)

I, Sam Malek, Ph.D., declare as follows:

1. I have been engaged by King & Spalding LLP on behalf of OpenText Inc. (“Patent Owner”), to provide this Declaration concerning the technical subject matter relevant to the petition for Post-Grant Review (“Petition”) of U.S. Patent 11,409,869 (“the ’869 patent”).

2. I am over 18 years of age. I have personal knowledge of the facts stated in this Declaration and could testify competently to them if asked to do so.

3. I have reviewed and am familiar with the specification and the claims of the ’869 patent. In general, I will cite to the specification of a United States patent using the following formats: (Exhibit No. Col:Line Number(s)) or (Exhibit No. Paragraph Number(s)). For example, the citation (EX1001 1:1–10) points to the ’869 patent specification at column 1, lines 1–10. Also, for convenience, I use italics to denote limitations from the challenged claims.

4. All the opinions in this Declaration are based on the documents I reviewed and my knowledge, experience, training and professional judgment. In forming the opinions expressed in this Declaration, I reviewed the Petition for Inter Partes Review of U.S. Patent No. 11,409,869 and supporting exhibits.

5. I have been asked to provide my technical opinions regarding how a person of ordinary skill in the art would have understood the claims of the ’869 patent at the time of the alleged invention. I have also been asked to provide my

technical opinions on how concepts in the '869 patent specification relate to claim limitations of the '869 patent. In reaching the opinions provided herein, I have considered the '869 patent, its prosecution history, and the references cited herein and have drawn as appropriate on my own education, training, research, knowledge, and personal and professional experience.

I. QUALIFICATIONS

6. In formulating my opinions, I have relied on my knowledge, training, and experience in the relevant field, which I will summarize briefly. A more detailed summary of my background, experience, and publications is set forth in my curriculum vitae (CV), which is submitted as Exhibit 2002.

7. I have more than two decades of experience in the general area of computer and software technology in a variety of roles, including as a software engineer, software architect, and programmer at a variety of companies, as well as a researcher and professor at several universities.

8. I am currently a Full Professor of Software Engineering with tenure in the Informatics Department within the School of Information and Computer Sciences at the University of California, Irvine. I am also the Director of the Software Engineering and Analysis Laboratory ("SEAL"). My general research interests are in the field of software engineering, and to date my focus has spanned the areas of software analysis and testing, mobile computing, architecture and design, and

software dependability and security. The underlying theme of my research has been to devise techniques and tools that aid with the construction, analysis, and maintenance of largescale software systems.

9. I received my Ph.D. and M.S. degrees in Computer Science from the University of Southern California and my B.S. degree in Information and Computer Science from the University of California, Irvine. I have received numerous awards for my research contributions, including the Test of Time Award from the Association for Computing Machinery Special Interest Group on Software Engineering (2020), National Science Foundation CAREER award (2013), GMU Emerging Researcher/Scholar/Creator award (2013), and the GMU Computer Science Department Outstanding Faculty Research Award (2011).

10. I have authored over 120 peer-reviewed research publications. These publications have been highly cited (close to 10,000 times according to scholar.google.com). I have received numerous research grants from a variety of organizations, including the National Science Foundation (NSF), Federal Bureau of Investigation (FBI), Air Force Office of Scientific Research, Department of Homeland Security (DHS), National Security Agency (NSA), Defense Advanced Research Projects Agency (DARPA), Office of the Director of National Intelligence, and Army Research Office among others. Software systems and results produced in

my research lab have been adopted for use by various organizations in both private (e.g., Boeing, Bosch) and public (e.g., NASA, DHS, FBI) organizations.

11. I also have several years of experience as a software engineer, software architect, and programmer at a variety of companies, namely the Boeing Company, PricewaterhouseCoopers Consulting (later acquired by International Business Machines (“IBM”) Corporation), FieldCentrix, and Neural Computing Systems Lab. During my tenure in the industry, I gained professional experience in the design, development, and management of largescale software engineering efforts. Among the software systems that I designed and developed were commercial ecommerce and enterprise software systems. An important part of my responsibility in these projects was implementation and evaluation of security mechanisms to protect the resulting software systems from security attacks.

12. I have taught software engineering and computer science courses on a regular basis at the B.S., M.S., and Ph.D. levels at the universities where I have been employed. I have been a frequent speaker at scientific conferences, symposia, workshops, and gatherings of software engineering academics and professionals. I have served as chair, committee member, and reviewer for numerous software engineering journals, magazines, and conferences. I am currently on the editorial board of the ACM Transactions on Software Engineering and Methodology, the ACM Transactions on Autonomous and Adaptive Systems, and the Springer Journal of

Computing. I am a member of the Association for Computing Machinery (ACM), ACM Special Interest Group on Software Engineering (SIGSOFT), and the Institute of Electrical and Electronics Engineers (IEEE). I have testified at trial and deposition as an expert witness in numerous cases in the past five years.

13. My experience is highly relevant to the technologies at issue in this case and their application here. As set forth above, during my tenure in industry, I contributed to the design and development of several largescale enterprise and e-commerce software systems. In most of these systems, security and integrity of the system was of great concern. I contributed to the design and construction of proper security methodologies, such as the implementation of access control policies, in these systems. After this experience, as part of my research at the university over the past 15 years, I have conducted original research and published peerreviewed manuscripts related to software security. Some of this work has involved the use of static and dynamic program analysis techniques together with machine learning for detection of security threats. For example, in a paper titled “Lightweight, ObfuscationResilient Detection and Family Identification of Android Malware,” published in the ACM Transactions on Software Engineering and Methodology in 2018, I described a tool called RevealDroid that statically extracts information from Android apps to train a machine learning model to distinguish between malware and benign apps. RevealDroid was adopted and used by the Department of Homeland Security

in their Software Assurance Marketplace platform for vetting software products used in the government. My work in this area was sponsored by more than 2 million dollars in research funding from the Department of Homeland Security, the National Security Agency, the Office of Director of National Intelligence, Defense Advanced Research Projects Agency, and the Federal Bureau of Investigation.

14. As part of the courses that I regularly teach at the University at both undergraduate and graduate levels, I cover topics that are directly related to the technologies at issue in this case including software security, security vulnerabilities and exploits, program analysis, artificial intelligence, and machine learning. As associate editor of several journals and committee member of many conferences, I regularly evaluate the novelty of article submissions related to the technologies at issue in this case (e.g., software security, program analysis, machine learning) for publication in these venues.

II. UNDERSTANDING OF RELEVANT LEGAL PRINCIPLES

15. I am not a lawyer, and I will not provide any legal opinions. Although I am not a lawyer, I have been advised certain legal standards are to be applied by technical experts in forming opinions regarding the meaning and validity of patent claims.

16. I understand that a patent claim is invalid if it is anticipated or obvious in view of the prior art, and that a claim can be unpatentable even if all of the

requirements of the claim cannot be found in a single priorart reference. I further understand that invalidity of a claim requires that the claim be anticipated or obvious from the perspective of a person of ordinary skill in the art at the time the invention was made.

17. I have been informed that a patent claim is invalid if it would have been obvious to a person of ordinary skill in the art. In analyzing the obviousness of a claim, I understand the following factors may be taken into account: (1) the scope and content of the prior art; (2) the differences between the prior art and the claims; (3) the level of ordinary skill in the art; and (4) any so called “secondary considerations” of nonobviousness if they are present.

18. I have been informed that a claim can be obvious in light of a single prior art reference or multiple prior art references. I understand that to prove that prior art or a combination of prior art renders a patent obvious, it is necessary to:

- 1) Identify the particular references that, singly or in combination, make the patent obvious;
- 2) Specifically identify which elements of the patent claim appear in each of the asserted references; and
- 3) Explain why a person of ordinary skill in the art would have combined the references, and how they would have done so, to create the inventions claimed in the patent. I further understand that exemplary rationales that may support a conclusion of obviousness include:

- Combining prior art elements according to known methods to yield predictable results;
- Simple substitution of known element for another to obtain predictable results;
- Use of known technique(s) to improve similar devices (methods or products) in the same way;
- Applying a known technique to a known device (method or product) ready for improvement to yield predictable results;
- “Obvious to try” – choosing from a finite number of identified, predictable solutions with a reasonable expectation of success;
- Known work in one field of endeavor may prompt variations of the work for use in either the same field or a different field based on design incentives or other market forces if the variations are predictable to a person of ordinary skill in the art; and
- Some teaching, suggestion, or motivation in the prior art that would have led a person of ordinary skill in the art to modify the prior art reference or to combine the prior art reference teachings to arrive at the claimed invention.

19. I have been informed that, in considering obviousness, hindsight reasoning derived from the patentatissue may not be used.

III. STATE OF THE PRIOR ART

20. At the time of the '869 patent, conventional approaches for diagnosing potential malware threats were costly and time consuming, making it difficult to

realistically address zero-day threats for all of the files entering a system. One of these approaches, which may be described as “static code-based analysis,” “typically focus[ed] on finding [known] malicious code blocks within a file.” (EX-1001 2:23-25). The drawbacks of codebased analysis are twofold: first, because it could only identify code blocks in an executable that were already known to be malicious, it could not identify a new threat; second, this approach was slow and resource intensive, requiring substantial investments in time and computational resources to disassemble the code and, if the executable was encrypted, to decrypt it as well.

21. One potential solution to the first problem (i.e., identifying malicious code that had not been seen before) is to use a “machine learning classifier.” (EX-1001 2:14-53.) Machine learning classifiers are software programs that are “trained” to recognize, and classify, the malicious or benign nature of new, unknown executables. They do so by training to recognize certain indicia of maliciousness or benignness, known as “features,” within a large group of executables whose degree of maliciousness is already known (*e.g.*, “known malicious executable files, known benign executable files, and [known] potentially unwanted executable files”). (*Id.*) This process is known as the “training” phase.

22. Based on these identified features, machine learning systems build a “classifier”—i.e., a predictive model that accepts a set of features as input and outputs a predicted classification. Once trained, this model can be used to classify

previously unseen executables: the security software extracts from the previously unseen executable the features that the classifier has been trained to recognize, inputs those features into the classifier, and obtains the predicted classification for the executable. These are the steps of the “classification stage.”

23. One exemplary machine learning classifier is known as a support vector machine (“SVM”). A SVM is typically trained by extracting a set of features (represented as a fixed-length “feature vector”) from each training sample and then finding a mathematical function that best separates the feature vectors into respective categories. To classify a new sample, the fixed set of features is extracted from the new sample (a process known as “feature extraction”), the features are used to create an input feature vector with the number of elements required by the SVM, and the feature vector is input into the SVM for classification. Other machine learning classifiers, such as artificial neural networks, process feature vectors similarly to obtain a resultant classification.

24. Because the computational time and resources required grows with the number of features a classifier has been trained to consider, the training phase sometimes includes a process known as “feature selection,” which is synonymous with the phrase “feature pruning.” Feature selection is the process of identifying the features that are the most empirically relevant to a determination of maliciousness, and configuring the classifier to consider only those features. Feature selection at the

training stage reduces the computational burden of the classification stage, as the classifier will need to extract fewer features and/or perform less complex mathematical computations.

IV. THE '869 PATENT

A. Summary Of The '869 Patent

25. The '869 patent improves upon the prior art by introducing a technique of performing feature selection at the classification stage (rather than merely during the training stage) and for doing so based on the values of static data points extracted from an executable at that stage. Performing feature selection during the classification stage allows the system to dynamically calibrate a system's performance based on the content of the executable being evaluated, turning "off" features that are unlikely to be valuable for classification of this executable. Doing so accelerates the performance of the classifier by not forcing it to determine a value for features unlikely to facilitate a positive categorization of the executable. It thus improves upon prior art "static" evaluation, whereby the classifier had to always evaluate, or "test," every feature predetermined during the training stage.

26. In the "training stage," the system disclosed in the '869 patent examines a training set of executables known to be malicious, benign, or potentially unwanted, and identifies a combination of features within those executables that function as indicia of the category with which they are correlated (i.e. features designating an

executable as malicious, benign, or potentially unwanted). Using these identified feature sets, the system generates a “classifier:” a predictive model that extracts features from a new, previously unknown executable, and outputs a category for that executable.

27. The '869 patent achieves this result by turning features “on” or “off” during the classification stage based on the content of the executable being evaluated. (EX-1001 7:43-8:7; 12:61-13:39.)

28. For example, the classifier may consider certain features (at the classification stage) only if the size of the executable being evaluated is above a given threshold or only if the executable includes a certain type of content (e.g., legal copyright information), or a particular permutation of that content’s subtypes (e.g. for legal information, “timestamp data,” “licensing information,” “copyright information,”). (*Id.* 7:43-8:7.)

29. To decide which features to turn on or off, the '869 Patent teaches analyzing the values of the “static data points” in the executable. A “static data point” may be a “point of reference,” such as the executable’s file size, section names, legal information, and so forth. (*See* EX-1001 5:61-6:57, 8:35-42.)

30. The specification provides many exemplary embodiments of static data points (EX-1001 Table 1.1, 8:35-42), and a POSITA would recognize that many of these embodiments correspond to fields defined by the “Portable Executable” file

format, such as “file size,” “PE header time,” and “linker version.” (See EX-1001 3:43-54, 4:29-31 (describing extraction of static data points from PE files). In such embodiments, a static data point may include a field name (i.e., a predefined character string such as “File Size”) and a field value (*e.g.*, a number such as 4982784).) (EX-1001 8:39.)

31. The claimed system can turn features on or off during the classification phase based on the values of the static data points extracted from the executable being classified. (EX-1001 5:61-6:57, 8:35-42.) For example, if the value of the “File Size” static data point is greater than a certain numerical threshold, then the classifier may determine values for fewer or additional features, in order to make a positive decision about whether the executable is malicious, benign or potentially unwanted. (*Id.*) The set of features ultimately extracted from the executable being classified may be used to create one or more feature vectors that are input into the appropriate classifier. (*Id.* 8:43-9:54.)

32. Claim 1 recites a method that corresponds to the classification stage of the invented technique:

- identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;
- 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;

identifying an executable file to be evaluated;

extracting, by the knowledge module, a plurality of static data points from the identified executable file;

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,

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.

(EX-1001 (claim 1).)

33. The claimed method begins with a *training stage*: identifies static data points and associates them with categories of executables—*e.g.*, “harmful” or “benign.” The method then proceeds to a *classification stage*. In that stage, the method calls for “identifying an executable file to be evaluated,” extracting “static data points from the identified executable,” and “generating a feature vector from ... [those] static data points.” The feature vector is generated using a classifier that was previously “trained”—*i.e.*, at the training stage—to classify the static data points based on training data comprising files known to fit into one of the ... categories of files.” One or more features of the feature vector are then “turned on or off” based

on whether a value of at least one of the static data points extracted from the executable being evaluated is within a predetermined range. For example, the classifier might turn certain features “off” if the “File Size” of the executable is smaller than some predefined threshold. Finally, the classifier classifies the executable by using support vector processing on the resulting feature vector.

34. Independent claims 10, 15, 18, 32, and 46 recite similar systems and methods that turn features of a feature vector on or off at the classification stage, based on the value of static data points extracted from the executable being evaluated. The claims differ in the final, “evaluating” step. For example, claims 10 and 15 replace the final evaluating step with the step of “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.” In contrast, the final step of claims 18, 32, and 46 is “evaluating the feature vector using a machine learning model to determine whether the executable fits into one of the categories of files.”

B. Claim Construction

35. I understand that the the parties have agreed 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. (Petition 7-10.)

36. Consistent with that understanding, I apply the plain and ordinary meaning of all claim terms.

C. Level Of Ordinary Skill In The Art

37. I understand that certain issues relating to the validity of the '869 patent must be judged from the perspective of a person of ordinary skill in the relevant art, as I discuss below. I have been asked to define the level of a “person having ordinary skill in the art” or “POSITA” that someone would have had at the time the alleged invention as claimed was made. In deciding the level of ordinary skill, I have considered the following factors, which I have been informed are relevant in determining the level of ordinary skill in the art:

- The levels of education and experience of persons working in the field;
- The types of problems encountered in the art;
- The prior art solutions to these problems;
- The rapidity with which innovations are made; and
- The sophistication of the technology.

38. I understand that Petitioner has defined the level of ordinary skill in the art as:

A person of ordinary skill in the art (“POSITA”) on May 12, 2015, 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. In the alternative, a POSITA could have two or more years of work experience relating to static data analysis and malware detection in lieu of the education requirements.

(Petition 10).

39. I agree with the Petitioner's definition of a POSITA.

40. I was at least a person of ordinary skill in the art at the time of the invention of the '869 patent.

V. OVERVIEW OF PRIOR ART

A. Overview Of Tahan

41. Tahan is an academic paper that discusses various machine learning approaches for detecting malware in executables. (EX-1005 at 1.)

42. Tahan discloses three different approaches to malware classification. First, Tahan describes the state of the prior art in machine learning (ML) malware classifiers. (EX-1005 at 1-5.) Second, Tahan proposes a new system called "Mal-ID basic." (Id. at 5-11.) Third, it discloses a variation called "Mal-ID with ML Generated Models." (Id. at 11-13.)

43. These three disclosures describe three different systems. The Petition constantly intermixes the elements of these three different systems in incompatible ways, without even acknowledging that the elements are being pulled from different systems, let alone explaining how the elements would be combined or why. In my

opinion, the intermixing of elements in this haphazard and unexplained way renders the Petition unclear and frequently nonsensical.

44. For the sake of clarity, I describe separately each of the three systems disclosed in Tahan that Petitioner relies on.

1. Background Disclosure

45. Tahan begins with an “Introduction” (EX-1005 at 1-5), in which it discusses the general idea of “using machine learning (ML) techniques to produce a binary classifier that is able to distinguish malware from benign files.” (Id. at 2.)

46. The Introduction’s description of that general idea matches the description of the prior art “training stage” I have provided above. See *supra* Section III (“State of the Art”). Specifically, Tahan describes a process in which: (1) features are extracted from executables in a training set, (2) features are selected, such that “redundant and irrelevant features are discarded,” and (3) “[t]he last phase is creating a classifier using the reduced feature vectors created in phase 2.” (Id.) In other words, the Introduction describes feature extraction and selection at the training stage.

47. The Introduction does not describe the classification stage in any meaningful detail. It simply notes that the classifier created during the training stage can be of any type, and it surveys the efforts of other researchers to use different types of features and classifiers. (Id.)

48. Thus, Tahan’s Introduction merely provides a background discussion of performing feature extraction, feature selection, and classifier training at the training stage—the same process I have described above as prior art. See *supra* Section III (“State of the Art”).

2. Mal-ID Basic

49. Following the Introduction, Tahan introduces a new machine learning system called “Mal-ID.” (EX-1005 at 5-13.) That system is fundamentally different from those described in the Introduction.

50. Tahan teaches that the “basic insight” that motivated Mal-ID is that “both legitimate developers and hackers” use the same “higher level development platforms” and “the same specialized libraries.” (Id. at 5.) Consequently, the “main idea” of Mal-ID is “to detect the file segments that originated from the development platform or from a benign third-party library (termed here the Common Function) and then disregard those segments” while “the remaining segments would be compared to determine their degree of resemblance to a collection of known malware.” (Id. at 7.) Tahan calls this process “common segment analysis.” (Id.)

51. The first variation of Mal-ID that Tahan discloses is called “Mal-ID basic.” (Id. at 9-11.)

52. Mal-ID basic proceeds in two phases: (1) a “setup phase,” in which data is gathered from a training set of executables, and (2) a “detection phase,” in which

“a previously unseen” executable is analyzed and classified using the data gathered in the setup phase. (Id. at 7.)

a. Setup Phase

53. In the setup phase, a training set of known executables is analyzed to determine (1) the Common Functions (i.e., those found in benign executables) and (2) the “Threat Functions” (i.e., those found in malicious executables). (Id.)

54. The system starts by “collecting two kinds of files: benign and malware.” (EX-1005 at 7.) Then, a “CFL repository [containing the Common Function segments] is constructed from benign files and [a] TFL repository [containing the Threat Function segments] is constructed from malware files.” (Id.) “The CFL is constructed first and then the TFL.” (Id. at 9.)

55. To construct the CFL, each benign “file is divided into 64 equal length segments” (id.), which are each further “broken into 3-grams (three consecutive bytes)” (Id. at 7.) Moreover, “[t]he training files are randomly divided into 64 groups.” (Id.)

56. Then, for every combination of 3-gram and file group, the system records in the CFL (1) whether the 3-gram appears in the files of that group (“3-gram-files-association”) and (2) which segment of the file the 3-gram appears in (“3-gram-relative-position-within-file”). (EX-1005 at 8-9.)

57. The TFL is then constructed: “Each file from the malware collection is broken into ... 64-byte segments,” “[e]ach segment is broken into 3-grams and then tested against the CFL” such that “[s]egments that are not in the CFL are added to the TFL.” (Id. at 9.) “[T]he end result is the TFL, a repository made of segments found only in malware and not in benign files.” (Id.)

b. Detection Phase

58. In the detection phase, the data collected in the setup phase is analyzed to classify a previously unseen executable. (EX-1005 at 9-10.)

59. At a high level, the classification algorithm simply (1) “[d]ivide[s] file F into S segments of length L” and (2) counts how many of those segments meet a given set of criteria. (Id. at 10.) That set of criteria is:

- **Unique Segment.** “all duplicated segments are removed”
- **Entropy Is Within Range.** “if the entropy is smaller than EntropyLow threshold or entropy is larger than Entropy-High ... then discard the segment.”
“Entropy” refers to “the entropy measure of the bytes within a specific segment candidate.”
- **Not in CFL.** “if the CLF-MFG [(i.e., maximum total number of file-groups in CFL that contain the segment)] is larger than zero, then discard the segment”
- **In TFL.** “if the TFL-MFG index [(i.e., maximum total number of file groups in TFL that contain the segment)] is smaller or equal to the ThreatThreshold”

... then discard the segment.” For example, “only segments that appear two times or more [in the TFL] remain.”

- **TFL-Spread Within Range.** “If the TFL-Spread index [(i.e., the spread of the signature’s 3-grams along the various areas for all the files in the TFL)] equals zero or if it is larger than what we term SR threshold, then discard the segment.”
- **Second Level Index Aggregation.** The segment must be “found in malware and not in the CFL.”

(EX-1005 at 10-11.)

60. To decide whether the executable is malicious or benign, the system simply counts how many of the executable’s segments meet the above set of criteria. (Id. at 11.) “If there are at least X segments” that meet the above set of criteria, the system determines that “the file is malware,” and “otherwise, [it] consider[s] the file as benign.” (Id.)

61. Tahan describes the executable’s 3-grams as the extracted “features,” and the “entropy,” “MFG,” and “Spread” measures calculated based on those 3-grams as “meta-features.” (EX-1005 at 9) (“Note that the features, as described above, are in fact meta-features as they are used to represent features of features (features of the basic 3-grams)”.)

62. As this description shows, the “feature selection” described in Tahan’s Introduction is fundamentally different from the process performed by Mal-ID: the feature selection during the training phase of the Introduction determines which features will be extracted at the classification stage, whereas the setup phase of Mal-ID determines parameters that will be used to analyze the features (i.e., 3-grams) that will be extracted at the later “detection phase.” In other words, the features that Mal-ID extracts at the “classification stage” (i.e., the “detection phase”) is always the same: “S segments of length L,” regardless of what was done at the “training stage” (i.e., the “setup phase”). (EX-1005 at 10.) The setup phase of Mal-ID thus does not perform “feature selection,” as described in the Introduction.

63. Moreover, this description also shows that the segments that each file is sub-divided into do not “predefined character strings,” and are identical to the random, fixed-length file chunks used in wavelet decomposition, a method that the applicants explicitly distinguished their invention from, to the assent of the Patent Office.

3. Mal-ID With ML

64. In Section 2.4, Tahan suggests “improv[ing] the Mal-ID basic method by using Mal-ID features with various classifiers, but instead of using the Mal-ID decision model ... we let various ML algorithms build the model.” (EX-1005 at 11.)

a. Setup Phase

65. The setup phase of Mal-ID with ML starts like that of Mal-ID basic: by building the CFL and TFL in the same manner as did Mal-ID basic. (*Id.* (“obtain a collection of segments for both the CFL and the TFL as explained in Section 2”).)

66. The setup phase of Mal-ID with ML differs from that of Mal-ID basic in that Mal-ID with ML goes on to (1) create a fixed-length feature vector to represent each training file; and (2) use the collection of feature vectors to train a machine learning classifier. (EX-1005 at 11-13.)

67. Mal-ID with ML creates the feature vectors as follows:

- Calculate the meta-features for each segment of each file. (EX 1005 11 (“For each file’s segment, we calculate the CFL-MFG, TFL-MFG and the TFL-spread based on the CFL and TFL,” and “[t]he measure is calculated as well.”).)
- Discretize the values of the meta-features by grouping them into a fixed number of bins. (*Id.*) (“We discretize the numeric domain of the above features” into “sub-domains (bins)”).
- For each file, create a feature vector indicating the number of the file’s segments associated with each bin. (*Id.*) (“For each file, count the number of segments associated with each bin”)

68. The result is that each executable in the training set is represented by a respective feature vector with the same number of elements (i.e., the number of meta-feature/bin combinations).

69. Mal-ID with ML then uses the extracted feature vectors to train a machine learning classifier. (EX-1005 at 12.) (“An induction algorithm is trained over the training set to generate a classifier.”). Tahan discloses that different types of classifiers can be used, and reports that the authors investigated using three particular types of classifiers: a single decision tree (“C4.5”), a Rotation Forest (“RF”), and a neural network (“NN”).

b. Detection Phase

70. At the detection phase (i.e., the classification stage), the Mal-ID with ML receives a new executable to classify, extracts from the executable the same set of features that were extracted from the training set and used to train the classifier at the setup phase, and inputs that feature vector into the trained classifier for classification. (EX 1005 at 13.)

71. Mal-ID with ML thus operates similarly to the traditional ML methods described in the Introduction: (1) a training stage selects the set of features and trains a classifier on that set of features, and (2) a subsequent classification stage extracts that predetermined set of features from the executable being classified and inputs those features into the trained classifier for classification.

B. Overview Of Menahem

72. Menahem discloses ideas for combining multiple malware classifiers into an “ensemble.” (EX-1006 at 1.)

73. Like Tahan, Menahem starts with an Introduction describing the general process of “detecting malicious executables using Machine Learning (ML) techniques.” (Id.) As described repeatedly above, that process involves (1) a training stage in which “[m]alicious and benign files are represented by a vector of features” that “are used to train a classifier,” and (2) a classification stage (which Menahem calls the “detection phase”) in which “an unknown file can be classified” using the trained classifier. (Id.) Menahem teaches that some classifiers can use features “extracted from the PE header” of executables. (Id.)

74. Menahem observes that, because different classifiers are built using different “inducer algorithms” and different “types of features,” “each classifier will perform differently,” such that some classifiers may be better or worse at classifying certain types of malware. (EX-1006 at 2.)

75. Menahem suggests that this problem can be solved by combining multiple types of classifiers into “multi-inducer ensembles.” (Id.) “The main idea of an ensemble methodology is to combine a set of classifiers in order to obtain a better composite global classifier.” (Id.) For example, Menahem discloses a “Majority Voting” scheme in which a group of different classifiers “vote” and the classification

chosen is “the class that obtains the highest number of votes.” (EX-1006 at 3.) Menahem also proposes more complex ways of combining the classifiers, including techniques called “stacking” and “Troika.” (EX-1006 at 4.)

76. Menahem teaches that, “to make the ensemble more effective, there should be some sort of diversity between the classifiers,” and that this “diversity is obtained by using different types of inducers.” (EX-1006 at 3.)

C. Overview Of Forman

77. Forman discloses a general-purpose machine-learning classification technique that uses what the reference calls “feature scaling.” (EX-1018 1:1-3.)

78. As shown in Figure 1 below, Forman’s technique follows the conventional machine learning pattern of a “training” stage (elements 12, 14, and 16) followed by a “classification” stage (elements 22, 24, 18, and 27).

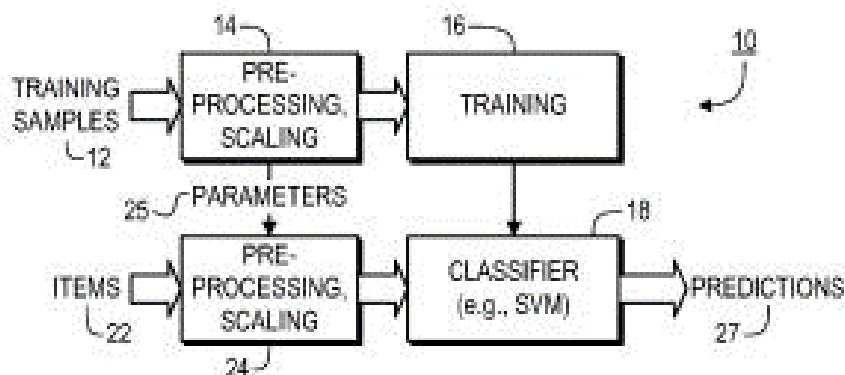


FIG. 1

(*id.* at Fig. 1.)

79. Forman’s classification system starts with a training stage. (EX1018 2:27-3:55 and 6:49-7:77.) The training stage begins by extracting feature vectors from the received “training samples” and then “pre-processing” those feature vectors. (*Id.* at 3:11-42.) The “preprocessing” can include traditional “feature selection” whereby features that are not sufficiently “predictive”—*i.e.*, have a low predictiveness “score”—are pruned. (*Id.* at 3:11-42.) Forman adds that the “preprocessing” can also include “scaling,” whereby the value of each feature is modified based, in part, the feature’s empirically or statistically determined predictiveness score. (*Id.* at 7:8-38.) The preprocessed (*i.e.*, pruned and scaled) feature vectors are then used to train a classifier. (*Id.* at 3:67-4:3.)

80. Forman’s system then performs a classification stage. (EX1018 3:56-4:3, 7:56-8:31.) In that stage, the system extracts a feature vector from the executable being classified and performs the same feature-selection and scaling that was performed during the training stage. (*Id.*) Importantly, the classification-stage feature-selection simply selects the *same* “feature set 38” that was selected at the training stage. Indeed, Forman’s specification is explicit that the item to be classified, *i.e.* “item 22,” is “represented by a set of values corresponding to a *feature set 38*”---the same set of features that were identified during the training stage. (*Id.* at 7:62-63) (emphasis added).

81. Saliently, moreover, Forman’s system does not prune features based on the values of the features extracted from the executable being classified; instead, it does so based on the feature’s empirically or statistically determined “predictiveness.” (*id.* at 7:8-38.). Forman’s system then scales the values of the remaining features, yielding results identical to those of the training stage. (*id.*). The preprocessed feature vector for the executable being classified is then fed into the classifier for classification.

VI. THE CITED ART DID NOT DISCLOSE OR RENDER OBVIOUS THE CHALLENGED CLAIMS.

82. I understand that The Petitioner and its expert allege that every limitation of Claims 1-4, 8, 10-25, 29, 32-39, 43, 46-53, and 57 obvious are rendered obvious by the combination of Tahan, Menahem and Forman, and that claims 5-7, 9, 26-28, 30-31, 40-42, 44-45, 54-56, and 58-59 are rendered obvious by the combination of Tahan, Menahem and Schultz.

Ground	Claims	Argument
1	1-4, 8, 10-25, 29, 32-39, 43, 46-53, and 57	§ 103(a): The combination of Tahan, Menahem and Forman render the claims obvious.
2	5-7, 9, 26-28, 30-31, 40-42, 44-45, 54-56, and 58-59	§ 103(a): The combination of Tahan, Menahem and Schultz render the claims obvious

83. I disagree with the arguments in the Petition and the opinions set forth by Petitioner’s expert for at least the reasons that I discuss in further detail below.

A. Ground 1: The Combination Of Tahan, Menahem, And Forman Did Not Render Claims 1-4, 8, 10-25, 29, 32-39, 43, 46-53, and 57 Obvious.

84. In my opinion, the combination of Tahan, Menahem and Forman fails to render obvious any claims of the '869 patent for three main reasons: 1) the combination fails to disclose selectively turning on or off features of a feature vector at the *classification* stage (*i.e.*, at the time an executable is evaluated for maliciousness), rather than at the *training* stage (*i.e.*, when a classifier is “setup” or trained); 2) the combination fails to disclose doing so based on the **values** of static data points exgracted from the executable falling within a predetermined range; and 3) evaluating such a feature vector using a “support vector machine” (as claimed by claims 1, 10 and 15) or a “machine learning model” (as claimed by claims 18, 32, and 46).

1. The proposed combination fails to teach turning features on or off at the classification stage.

85. In my opinion, neither of Tahan, Menahem and Forman, alone or in combination, render obvious any claim of the '869 Patent.

i. Tahan

86. In my opinion, Petitioner is incorrect that Tahan teaches turning features “on or off” at the classification stage (*i.e.*, Tahan’s “Detection Phase”). (Petition 23-24.) As the Board itself recognized in the '869 Sophos Denial, a feature that is “turned ... off” “is not used in the [classifier’s] evaluation.” ('869 Sophos Denial 25.) However, Tahan’s “Detection Phase,” which Petitioner and its expert argue

discloses turning features off during the classification stage, calls for extracting and analyzing **every segment** (i.e. the purported “features”) of the executable in the same way. In other words, **every segment of an executable** is used to evaluate that executable. Tahan’s “Detection Phase” therefore does not turn “off” any features.

87. As I explain above, one crucial benefit of turning features “on or off” at the classification stage is to optimize the classifier’s performance (e.g., runtime and/or accuracy) based on the content of the executable being classified. By doing so, a classifier is not forced to extract and evaluate the full set of features that was predetermined at the training stage, without regard to the content of the executable being classified at the classification stage. (EX-2001, at ¶¶31-32.) Instead, the classifier can turn some features “off” and thus avoid the work of evaluating redundant features and potential false positives introduced by extracting and considering values for those features.

88. Tahan’s system, by contrast, lacks these benefits because it performs the same analysis, during classification, as was predetermined during training, **regardless of the content** of the executable being classified. Namely, Tahan’s classification stage (“Detection Phase”) extracts **every segment** in the executable and tests every one to determine if any segment satisfies a given set of criteria. (EX-1005 at 10-11.). If at least one segment meets the criteria, the executable is classified as malware. (Id.) If not, the executable is classified as benign. (Id.) Because every segment

is extracted and considered in the same way, no segment is “turned on or off.” In Tahan, ***all segments, and thus all features***, are “on.”

89. Petitioner’s argument relies on Tahan’s labeling of segments that fail to meet the criteria as “discarded.” (Petition 23; ’844 Trend Denial at 15-16.) As the Board mistakenly did in the ’844 Trend Denial, Petitioner equates labeling a segment “discarded” with turning it “off.” In my opinion, that mapping is wrong because segments that have been ***labeled*** “discarded” are nonetheless considered and evaluated by the classifier no less than segments that are not so labeled.

90. In other words, merely labeling a segment “discarded” does not exclude the segment from some further consideration that segments not labelled “discarded” undergo. As I explain above, ***every segment*** in Tahan is processed in precisely the same way, which results in receiving a label, e.g. “discarded” or not discarded. After all segments have been labeled, and thus necessarily after ***all segments have been processed and evaluated***, the classifier merely asks whether all segments received the “discarded” label. The classification decision thus turns equally on all segments. The segments labeled “discarded” are therefore not “turned ... off” in any logical sense. In my opinion, Tahan thus does not teach turning features off at the classification stage.

ii. Combination with Menahem

91. In my opinion, Petitioner’s proposed addition of Menahem to Tahan also does not remedy the shortcomings discussed above, and Petitioner makes no effort to argue otherwise. (Petition 24.) The petition devotes only one passing sentence to explain why combining Menahem with Tahan might suggest turning features on or off at the **classification stage**: namely, after noting that Tahan’s decisions to apply the “discarded” label to certain segments may involve considering the segment’s “gain ratio,” Petitioner notes that “Menahem describes similar compatible selection functionality.” (Petition 24.) However, to support that contention, Petitioner cites the expert declaration of Dr. Nielson, who opines merely that “Menahem also discloses feature selection during **training**,” including by considering “gain ratio.” (EX-1003 ¶¶ 277-82 (emphasis added).) Thus, Petitioner does not even argue, let alone show, that Menahem teaches feature selection at the **classification stage** or that combining Menahem with Tahan would render that functionality obvious.

92. Moreover, the portion of Menahem Dr. Neilson cites—Section 3 (EX-1006 at 6-7)— confirms his opinion that Menahem “discloses feature selection during **training**.” (EX-1003 ¶¶ 277-82 (emphasis added).) As the outline of Menahem’s Section 3 explains, Sections 3.2 and 3.3 of Menahem “describe the training process and the datasets we had used.” (EX-1006 at 5 (emphasis added).) Section 3.2 discloses “trimming the number of features” by “identify[ing] those terms that appear

most frequently in files” of the training set. (EX-1006 at 5.) Indeed, the disclosure of Section 3.2 precedes the description of “[t]he training process” in Section 3.3, which details “the training of the base-classifiers.” (EX-1006 at 6.) Thus, in my opinion, Section 3 of Menahem does not describe turning features on or off at the classification stage. (EX-2001 at ¶92.)

93. In my opinion, Petitioner does not, and cannot, argue that adding Menahem to Tahan would have rendered it obvious to turn features on or off at the classification stage

iii. Combination with Forman

94. In my opinion, Petitioner’s argument that the combination of Tahan with Forman’s teaching of “pre-processing” feature vectors teaches turning features “on” or “off” at the classification stage is also incorrect. (Petition 24-28.) Petitioner’s argument is that Forman’s classification-stage “pre-processing” can include feature selection, and that it would therefore have been obvious to modify Tahan to “discard” segments “also during classification.” (*Id.* at 25-26.).

95. In my opinion, argument fails for two main reasons. First, Petitioner’s proposed combination of Tahan with Forman addresses the wrong problem: the trouble with Tahan is not that the “discarded” labeling occurs outside the classification stage, but that the labeling (which does occur at the classification stage) does not

turn features on or off. Second, Petitioner is wrong that Forman teaches feature selection at the classification stage.

96. First, Petitioner’s proposed combination addresses the wrong problem with Tahan. Petitioner’s argument is that Forman would have made it obvious that Tahan’s labeling of a segment as “discarded” can occur at the classification stage. (Petition 25.) But the problem with Tahan is not that segments are labelled “discarded” outside the classification stage. In fact, the labels ***are applied*** at the classification stage. However, the real problem with Tahan’s disclosure, as I explain above, is that this labeling does not turn the segment “off” within the meaning of the claims. Thus, in my opinion, the proposed combination with Forman should be rejected because it fails to suggest how Tahan might be modified to turn features off at the classification stage.

97. Second, Petitioner is incorrect that Forman alone teaches turning features on or off at the classification stage. (Petition 25-26.) Instead, just like conventional machine learning systems, Forman identifies the “feature set” to be used at the training stage, based on which features are most “predictive” at that stage, and does not disclose a separate, independent feature selection process (or corresponding criteria) for the classification stage. (EX-1018 2:59-3:42; *see also id.* at 3:11-14, 6:52-7:7.)

98. For example, in Forman’s training stage, “training samples 12, initially are input into a pre-processing section 14,” and “[e]ach of the training samples preferably is represented by values for features in a designated feature set.” (*Id.* at 2:29-33; see also *id.* at 6:53-55.) The training stage then executes a “pre-processing” step 14 that performs feature selection, for example determining and pruning the least “predictive” features from the original designated feature set. (*Id.* at 2:27-3:42.) Finally, the “modified feature sets 38 are input into training section 16” to train the classifier. (*Id.* at 3:44-45.)

99. In Forman’s classification stage, the classifier uses exactly the **same** “feature set 38” identified in the training stage. For example, Forman discloses that the item to be classified (“item 22”) “is represented by a set of values corresponding to a feature set 38”—i.e., the **same** feature set identified in the training stage. (*Id.* at 7:62-63 (emphasis added).)

100. Although Forman does illustrate the classification stage as having its own “pre-processing” step 24, the role of that “pre-processing” step 24 is simply to extract the same feature set that was determined at the training stage. Indeed, Forman discloses that the “processing parameters” “generated in [training-stage] pre-processing section 14 [are] provided for use by pre-processing section 24,” which “performs processing that is identical or at least substantially similar to that of [training-stage] pre-processing section 14.” (*Id.* at 3:56-60.)

101. Thus, in my opinion, Forman’s classification-stage “pre-processing” step 24 is simply a conventional feature extraction step—that is, the process of extracting values for the same “feature set 38” identified at the training stage. The choice of which features to select (i.e., feature selection) is not made at the classification stage based on anything seen at that stage, but rather, at the training stage, based on empirically or statistically determined feature “predictiveness,” as is traditional.

102. Moreover, to the extent that Forman’s classification-stage “pre-processing” can be characterized as turning features on or off (it cannot), the Board should still reject Ground 1 for two reasons: (1) as I explain above, Petitioner has failed to articulate how the combination of Tahan and Forman would turn features on or off at the classification stage, and (2) I explain below, neither Forman’s nor Tahan’s purported turning of features “on or off” is based on the values of static data points extracted from the executable being classified.

2. The proposed combination fails to teach turning features on or off based on the values of static data points extracted from the executable being classified.

103. In my opinion, the proposed combination of Tahan, Menahem and Forman should be rejected for an additional ground: as the Board found in the ’844 Trend Denial, Tahan’s decision to label certain segments as “discarded” is not based on the *values of static data points* extracted from the executable being classified,

but rather, on the values of certain “*meta-features*” of the segments, such as entropy. (’844 Trend Denial 17.) In my opinion, neither Menahem nor Forman teach or suggest that the selection be based on the values of static data points extracted from the executable being classified. Indeed, Petitioner does not argue that either of those secondary references suggest that feature. Thus, in my opinion, the proposed combination fails to teach or suggest this limitation.

i. Tahan

104. Regarding Tahan, Petitioner argues that “Tahan teaches selecting segments (features / static data points) of files for the feature vector *based on an entropy or CFL-MFG index value of the segment.*” (Petition 23 (emphasis added).) However, as the Board found in the ’844 Trend Denial, a decision “based on entropy and the CFL-MFG, TFL-MFG, and TFL-Spread indices” is not a decision that is based on “the values of the segments themselves;” decisions based on entropy, the CFL-MFG, etc. are decisions based on the segments’ “meta-features.” (’844 Trend Denial at 17.) I agree with the Board’s reasoning in the Trend Denial: “Petitioner does not explain how or why the meta-features or the indices [in Tahan] should be considered ‘a value of one or more static data points,’” as claimed. (*Id.*) It is therefore also my opinion that the Board should reach the same conclusion here.

105. Petitioner also argues that Tahan’s decision to apply the “discarded” label to a segment may turn **on** the “gain ratio” of the segment (Petition 24.)

According to Petitioner, “gain ratio” is a value reflecting the degree to which “the additional segment adds any signal gain in helping [the classifier] identify malware or whether the segment is [instead] noise.” (*Id.*)

106. But Petitioner’s own description is of gain ratio as a measure of ***effectiveness***, not the value of the segment. Thus, in my opinion, Tahan’s use of a “gain ratio” does not, and cannot, show that Tahan turns any features “on or off” based on the ***value*** of a static data point extracted from the executable being classified. I understand that Board reached exactly that conclusion when it rejected Petitioner’s identical argument in the ’844 Trend Denial. (Trend Micro Inc., IPR2023-00662, Paper 1 at 21 (PTAB March 3, 2023) (Petitioner arguing that Tahan’s use of “gain ratio” teaches turning features on or off based on the values of static data points).

ii. Combination with Menahem

107. In my opinion, the addition of Menahem does not show that it would have been obvious to modify Tahan to turn features on or off based on the values of static data points extracted from the executable being classified. Moreover, Petitioner does not even make that argument. (Petition 24.)

108. As I explain above, Petitioner’s only use of Menahem to disclose the “turning on or off” of features is the observation that Menahem’s training-stage feature selection may consider a feature’s “gain ratio.” (Petition 24.) And as I explain in the immediately preceding section, Petitioner describes “gain ratio” as a measure

of a feature's classification *effectiveness*, not the value of a static data point. Thus, Petitioner does not argue that Menahem teaches feature selection based on the values of static data points extracted from the executable being classified.

109. Although the Petition does not discuss it, Petitioner's expert, Dr. Nielson, cites Menahem's use of "Inverse Document Frequency" ("TF-IDF") as constituting a value of training-stage feature selection. (EX-1003 ¶¶ 279, 280.) However, Dr. Nielson admits that "'TF-IDF values' represent values *derived from* n-grams," not the value of the n-grams themselves. (Id. ¶ 280)(emphasis added). Thus, as with Tahan's entropy values, Menahem's TF-IDF values are meta-features. Therefore, in my opinion, the Bour should, just as it held with respect to Tahan's "meta-features," also hold that Menahem's meta-features fail to satisfy the value limitation because "Petitioner does not explain how or why the meta-features ... should be considered 'a value of one or more static data points.'" ('844 Trend Denial 17.)

iii. Combination with Forman

110. Similarly, it is my opinion that the addition of Forman to the combination of Tahan and Menahem also fails to teach feature selection based on the value of static data points extracted from the executable being classified. Once again, Petitioner does not argue to the contrary. (Petition 24-25.)

111. Petitioner argues that Forman teaches classification-stage feature selection that is "identical or at least substantially similar to that of [training-stage] pre-

processing section 14.” (Petition 25 citing EX1003, ¶ 283 quoting EX1018, 3:56-60.) But Forman teaches that the “pre-processing section 14,” which occurs in the **training stage**, performs “feature selection” based on “scores” of how “predictive” the features are. (EX1018 2:59-3:42, 3:11-14, 6:52-7:7.) Those “scores,” which were “generated in pre-processing section 14[, are] provided for use by pre-processing section 24,” which occurs in the **classification stage** (*Id.* at 3:60-65.) Thus, Forman’s classification-stage “pre-processing” step 24 selects features based on **exactly the same** predictivness scores that were generated during training, i.e. as part of pre-processing section 24. It does not do so based on the values of static data points extracted from the sample being classified at the classification stage.

112. Indeed, Forman never suggests that the classification-stage “pre-processing section 24” selects features based on anything found in the executable being classified, let alone based on the values of static data points extracted from that executable. For example, Forman does not disclose any parameters used by classification-stage “pre-processing section 24” other than those received from training-stage “pre-processing section 14.”

113. Thus, even if Forman’s classification-stage “pre-processing” did constitute a disclosure turning features “on or off” (which it does not), Forman would still fail to teach doing that based on the values of static data points from the executable being classified. The addition of Forman to Tahan and Menahem therefore does

not teach or suggest turning features on or off based on the values of static data points extracted from the executable being classified at the classification stage.

3. The proposed combination fails to teach evaluation of the recited feature vectors using a “support vector machine” or any other “machine learning model.”

114. Finally, it is my opinion that the Board should deny institution based on a third, independent ground: as the Board found in the '844 Trend Denial, even if Tahan's marking of certain segments as “discarded” constitutes turning those segments “off” (it does not), and even if Tahan were to make those marking decisions based on the values of static data points extracted from the executable being classified (it does not), Tahan would still not teach evaluating the non-discarded segments using a “support vector machine” (as recited in claims 1, 10, and 15) or a “machine learning model” (as recited in claims 18, 32, and 46). ('844 Trend Denial 18-19.) As the Board found in the '844 Trend Denial, Tahan “uses a different method to evaluate the feature vector”—i.e., simply counting the number of non-“discarded” segments. ('844 Trend Denial 19.)

115. Thus, in my opinion Tahan fails to teach the final limitation of each independent claim, which requires providing the modified feature vector to a “support vector machine” or otherwise “evaluating” it using some other “machine learning model.” I understand that the Board has already held that combination with

Menahem fails to remedy these shortcomings (*id.* at 19); it should hold the same for the proposed further combination with Forman introduced here.

i. Tahan

116. In Tahan’s Mal-ID system, the purported “feature vector” whose features were purportedly “turned on or off” (i.e., labeled “discarded” or not) is not subsequently evaluated by an SVM or any other machine learning model, as recited by the claims. Instead, after the Mal-ID system decides which segments are “discarded” (i.e., purported turning features on or off), it makes its classification decision based simply on whether any segment was not “discarded,” rather than evaluating any feature vector with a machine learning model (EX-1005 10-11.) Determining that at least one segment is not “discarded”, as disclosed by Tahan, does not involve *any* machine learning model, let alone an SVM. Instead, it is a simple, hard-coded if-then statement based on no previous learning.

117. Petitioner argues in passing that it would have been obvious for Mal-ID’s classification decision to be made using an SVM or other machine learning model instead of the hard-coded if-then statement because “Tahan discloses that support vector machines (SVMs) are a known classification technique.” (Petition 28.) However, as the Board has already explained when it rejected Petitioner’s identical argument in the ’844 Trend Denial, “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.” (’844 Trend Denial 19.)

118. The Petition includes no new arguments regarding this issue with respect to Tahan. Thus, in my opinion the Board should again reject Petitioner’s unsubstantiated argument that the mere existence of SVMs and machine learning models in the prior art would have made it obvious to modify Mal-ID’s classification decision to replace its hard-coded if-then statement with some type of machine learning model.

ii. Combination with Menahem

119. Menahem fails to remedy the problems with Tahan regarding the use of an SVM or machine learning model. (Petition 28-29.) Petitioner relies on Menahem’s disclosure of SVMs generally and of a “composite classifier,” wherein the output prediction from a “base classifier” becomes the input into a separate, “final classifier.” (Petition 29.) However, Petitioner does not explain how these teachings might be combined to suggest that Tahan’s purported reduced feature vector (which is not Tahan’s final output) would become the input into an SVM or other machine learning model. The corresponding portions of Dr. Nielson’s declaration provide no additional clarity. (EX-1003 ¶¶ 296-300.)

120. I understand that Petitioner’s argument here appears to be the same one it made in its earlier, failed petition against the parent ’844 patent. (See *Trend Micro Inc. v. Webroot Inc.*, IPR2023-00662, Paper 1 at 51-53 (PTAB March 3, 2023). As in those proceedings, the Board should reject the argument because “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.” (’844 Trend Denial 19.)

121. To the extent that Petitioner is suggesting that it would have been obvious to create a “composite classifier” where the output of Tahan’s Mal-ID system is the input to an SVM or other machine learning model, this argument must fail because the output from Tahan’s Mal-ID system is a classification, not the purportedly pruned feature vector. As Petitioner’s expert observes (EX-1003 ¶ 296), Menahem’s idea is to “ensemble the results of multiple classifiers,” for example, using a “Majority Voting Schema.” Petitioner fails to point to any disclosure in Menahem that would have fa a POSITA to replace Tahan’s decision step with a step that instead passes an intermediate output (i.e., Tahan’s purportedly pruned feature vector) into a separate classifier (such as an SVM), let alone how that would work or why a POSIT would have had a reasonable expectation of success.

122. In my opinion, Board should reaffirm its previous holding that Petitioner has failed to explain how the combination of Menahem with Tahan would

have taught that the purported feature vector in Tahan should be provided to an SVM or otherwise evaluated by any other machine learning model.

iv. Combination with Forman

123. In my opinion, combining Forman with Menahem and Tahan does not resolve any of the shortcomings of Tahan and Menahem described above. Petitioner's only argument regarding Forman on this limitation is passing observation that Forman evaluates feature vectors using an SVM classifier. (Petition 30.) As the Board has already explained, however, "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." ('844 Trend Denial 19.) Thus, Petitioner's observation that Forman used SVM classifiers in the conventional way did not render it obvious to replace Tahan's evaluation method with an SVM, let alone explain how that replacement would operate.

124. For at least the foregoing reasons, it is my opinion that the combination of Tahan, Menahem, and Forman does not suggest evaluating the claimed feature vector using a "support vector machine" (as recited in claims 1, 10, and 15) or a "machine learning model" (as recited in claims 18, 32, and 46).

B. Ground 2: The combination of Tahan, Menahem, and Schultz Did Not Render Dependent Claims 5-7, 9, 26-28, 30-31, 40-42, 44-45, 54-56, and 58-59 Obvious.

125. In my opinion, Ground 2 fails for similar reasons as Ground 1. The Ground adds Schultz only to purportedly teach the limitations added by dependent claims 6, 7, 9, 15, 16, 18, and 20. (Petition at 55-69.) Because Ground 2 includes no argument that Schultz remedies any of the shortcomings of Ground 1 identified above with respect to the independent claims, and the claims Ground 2 challenges all depend on those independent claims, Ground 2 cannot render any of the challenged claims obvious.

VII. CLAIMS AND/OR LIMITATIONS NOT ADDRESSED IN THIS DECLARATION

126. It is my understanding that this declaration is being provided in support of Patent Owner's Preliminary Response. I have provided analysis of key limitations not included in the prior art. The fact that I have not analyzed particular limitations or claims in no way indicates or supports that these limitations or claims are disclosed or rendered obvious by the prior art, and I reserve the right to supplement my analysis if asked to do so.

VIII. SECONDARY OBJECTIVE INDICIA OF NONOBVIOUSNESS

127. I understand that in determining obviousness of the claimed invention, objective evidence of secondary considerations pertaining to nonobviousness must be considered, including longfelt but unsolved needs, failure of others, commercial

success, etc. (*In re Huang*, 100 F.3d 135, 139 (Fed. Cir. 1996) (citing *Graham v. John Deere Co.*, 383 U.S. 1, 1718 (1966).) I further understand that these indicia constitute independent evidence of nonobviousness that can be the most probative evidence in that regard “in avoiding the trap of hindsight when reviewing, what otherwise seems like, a combination of known elements.” (*Leo Pharm. Prods., Ltd. v. Rea*, 726 F.3d 1346, 1358 (Fed. Cir. 2013).)

128. As demonstrated by my summary of the prior art, above, there was at least a longfelt and persistent need prior to the '869 patent to improve upon prior art classification performance by calibrating feature vectors, at the classification stage, based on the content of the executable to be classified. (See Section III, *supra*.) Prior attempts to develop such methods were unable to perform such dynamic calibration at the classification stage. (See Section III, *supra*.) Where others failed, the '869 Patent succeeded. (See Section IV(A), *supra*.)

129. Thus, these secondary indicia only further support my opinion that the '869 Patent was nonobvious.

IX. CONCLUSION

130. For at least the foregoing reasons, it is my opinion that Petitioner's grounds do not render obvious any of the challenged claims of the '869 Patent.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true;

and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed this 21st day of December, 2023 in Irvine, CA.

Respectfully submitted,

A handwritten signature in cursive script, appearing to read "Sam Malek", written in black ink.

Sam Malek, Ph.D.