

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AO KASPERSKY LAB

Petitioner

v.

OPEN TEXT INC.

Patent Owner

Case No. IPR2023-01160

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 Ideses	17
	B. Overview Of Sharma.....	20
VI.	THE CITED ART DID NOT DISCLOSE OR RENDER OBVIOUS THE CHALLENGED CLAIMS.	22
	A. Ground 1: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52- 59 Were Not Obvious In View Of Ideses	22
	1. Ideses does not render independent claims 18, 32, or 46 obvious	23
	2. Ideses does not render any of the dependent claims obvious	26
	B. Ground 2: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52- 59 Were Not Obvious In View Of The Combination of Ideses and Sharma	26
	1. The proposed combination does not remedy the shortcomings of Ideses.....	27
	2. Sharma cannot remedy the shortcomings of Ideses.....	28

VII. CLAIMS AND/OR LIMITATIONS NOT ADDRESSED IN THIS DECLARATION	30
VIII. SECONDARY OBJECTIVE INDICIA OF NONOBVIOUSNESS	30
IX. CONCLUSION.....	31

EXHIBIT LIST

Exhibit	Description
2001	Declaration Of Sam Malek, Ph.D.
2002	<i>Curriculum Vitae</i> of Sam Malek, Ph.D.
2003	<i>Webroot, Inc., Open Text Inc., v. AO Kaspersky Lab et al.</i> , Case No. 6:22-cv-00243-ADA-DTG, Claim Construction Order, Dkt. 391 (W.D. Tex July 25, 2023)
2004	<i>Sonrai Memory Ltd. v. Kingston Tech. Co. and Kingston Tech. Corp.</i> , Case No. 6:21-cv-1284-ADA, Order Denying Kingston Tech. Co. and Kingston Tech. Corp.’s Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 94 (W.D. Tex., Oct. 18, 2022)
2005	<i>mCom IP, LLC v. Cisco Systems, Inc.</i> , Case No. 6:22-cv-00261-ADA, Order Denying Defendant’s Motion to Stay Pending <i>Inter Partes</i> Review, Dkt. 42 (W.D. Tex., Oct. 20, 2022)
2006	<i>Intellectual Ventures I LLC et al. v. Hewlett Packard Enterprise Co.</i> , No. 6:21-cv-00596-ADA, Order Denying Hewlett Packard Enterprise Company’s Opposed Motion to Stay Pending Resolution of <i>Inter Partes</i> Review, Dkt. 104 (W.D. Tex., Dec. 22, 2022)
2007	Email dated 11/8/2023 from David Holt to Trials@USPTO.gov re CrowdStrike v Open Text / Webroot IPRs: Request to File Motions to Terminate
2008	Email dated 11/13/2023 from Trials@USPTO.gov to David Holt RE: CrowdStrike v Open Text / Webroot IPRs: Request to File Motions to Terminate
2009	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Dec. 9, 2022 Transcript of Motions Hearing, Dkt. 133 (W.D. Tex. Dec. 15, 2022) (excerpted)
2010	Dani Kass, “Catching Up On Patent Litigation With Judge Albright” (Law360 Mar. 14, 2023);

Exhibit	Description
	https://www.law360.com/pulse/articles/1582438/print?section=pulse/courts
2011	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab. et al.</i> , Case No. 6:22-cv-243-ADA-DTG, Unopposed Motion to Modify the Scheduling Order, Dkt. 468 (W.D. Tex. Oct. 23, 2023)
2012	<i>Webroot, Inc., Open Text Inc., v. AO Kaspersky Lab et al.</i> , Case No. 6:22-cv-00243-ADA-DTG, Order Granting Fifth Amended Scheduling Order, Dkt. 252 (W.D. Tex., Mar. 29, 2023)
2013	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA-DTG, Defendant's AO Kaspersky Lab's Preliminary Invalidity Contentions, Appendix C (W.D. Tex., Feb. 21, 2023) (excerpted)
2014	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA-DTG, Defendant Sophos Ltd.'s Final Invalidity Contentions (W.D. Tex., Nov. 1, 2023) (excerpted)
2015	<i>Webroot, Inc. and Open Text Inc., v. AO Kaspersky Lab</i> , Case No. 6:22-cv-00243-ADA, Second Amended Complaint for Patent Infringement, Dkt. 122 (W.D. Tex., Dec. 9, 2022)
2016	<i>Webroot, Inc. and Open Text Inc., v. Forcepoint LLC</i> , Case No. 6:22-cv-00342-ADA, Defendant Forcepoint LLC's Final Invalidity, (W.D. Tex., Nov. 1, 2023) (excerpted)
2017	<i>Webroot, Inc. and Open Text Inc., v. Trend Micro Inc.</i> , Case No. 6:22-cv-00243-ADA, 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 Ideses

41. Ideses is directed to the generation of machine learning classifiers that are suitable for resource-constrained client devices, such as smartphones. (EX-1005 at [002]-[0006].)

42. Because of the computational complexity of the operations involved in machine learning classification and training, mobile devices, which are often "resource limited," Ideses discloses leveraging a "central server" to perform a substantial portion of such operations. (EX-1005 [0060].) In particular, Ideses discloses using this server to identify, at the training stage, an optimal, "pruned" set of features that are both computationally inexpensive to test and empirically effective at categorizing a given application as desirable or undesirable. (*Id.* [0061].) After this pruned set of features has been introduced into the mobile device,

“classification...may take place” on that device “during run-time” within the resource constraints of the device. *Id.*

43. For the sake of clarity, I have reproduced Figs. 1 and 2 of Ideses, which respectively demonstrate the “training stage” (Fig. 1) and the “classification stage” (Fig. 2) disclosed by Ideses, with an annotation (in RED) indicating at which stage feature selection occurs.

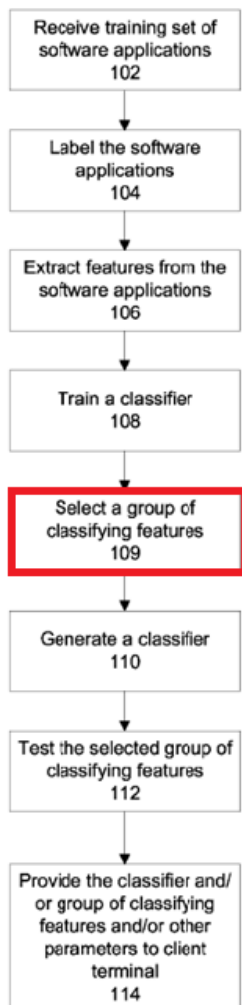


Fig. 1 (Training)

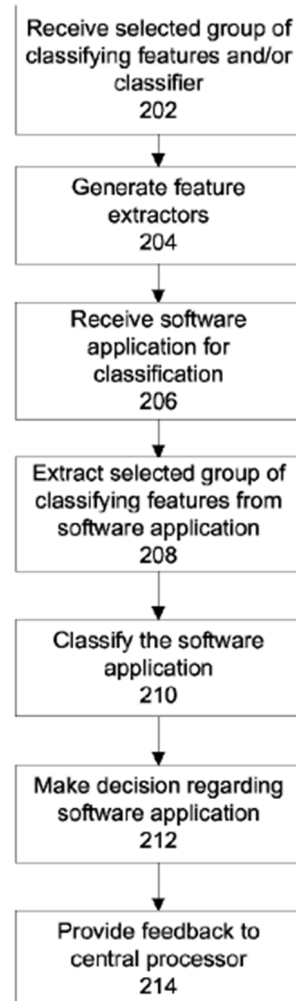


Fig. 2 (Classification)

(EX-1005 Figs. 1 and 2 (annotated).)

44. As shown in Fig. 1 above, feature selection in Ideses occurs at the training stage. This stage, as disclosed by Ideses, begins with the central server receiving a “training set of software applications” (step 102). That training set is then categorized (step 104), i.e. “Labeled,” as either “desired” or “undesired.” (EX-1005 [0081]-[0085].) This labeling can be performed either by human supervisors or through a “non-supervised”, *e.g.* algorithmic approach, that relies on drawing correlations between clusters of applications to determine which category each application belongs to. (*Id.* [0085].) In the next step (step 106), features are extracted from each categorized software application, and then (step 108) those features (and the categories with which they are associated) are used to train the classifier to recognize an application as desired or undesired. (*Id.* [0094]-[0096].)

45. At step 109, The training system disclosed by Ideses performs feature selection and feature “pruning”—i.e., the process of selecting the features that the trained classifier will consider (step 109). Ideses teaches that choosing the appropriate features most relevant to classification is based on two factors: 1) “identifying a classification effectiveness rank for each of the features” and/or 2) “resource requirements for each of the features.” (EX-1005 [0099] (emphasis added).) In other words, features are selected at the training stage by optimizing “the tradeoff” between “improving classification ability” and “reducing CPU requirements and/or ... memory usage.” (*Id.* [0113].)

46. By performing this optimization, “the group of classifying features is selected” to both achieve accurate classification and “to allow calculation and/or extraction of the features during run-time on the mobile device.” (*Id.* [0103].)

47. Figure 2 illustrates the classification stage disclosed by Idseses, as executed by a resource-constrained mobile device—i.e., “execution of a classifier based on a pruned feature list” that the mobile device can both extract and classify using its own resources. (EX-1005 [0123].) The mobile device: (1) receives the classifier and pruned feature list from the server (step 202); (2) generates or identifies a feature extractor for the pruned feature list (step 204); (3) receives a new executable to evaluate (step 206); (4) uses the extractors to extract the pruned feature list from the new executable (step 208); (5) inputs the extracted features into the classifier to classify the new executable (step 210); uses the output of the classifier to decide whether to execute the new executable (step 212); and (6) reports the outcome back to the server (step 214).

B. Overview Of Sharma

48. Sharma discloses a “risk-rating framework for mobile computer applications” or “RRF.” (EX-1006 at [0024].) The framework includes two stages: training (as depicted in Figure 3) and classification (as depicted in Figure 4).

49. Sharma discloses that a “Feature Module” is used to extract features, a “Training Module” is used to train a classifier on those features (or a selection

thereof), and a “Risk-Rating Module” uses the classifiers to classify new executables. (*Id.* at [0035]-[0036].)

50. Figure 3 shows a method “for training a RRF system.” (*Id.* at [0037].) The method therefore corresponds to the training stage. The Figure shows stages analogous to the usual stages involved in AI training. The process starts by “receiving a data corpus of mobile computer applications,” including “known-malware and known-benign computer applications.” (*Id.*) Then, “features are extracted from each of the computer applications.” (*Id.*) Finally, “the extracted features ... are used to train the RRF system.” (*Id.*) The training can include training a “Partial Least Square model classifier, or a Bayesian model classifier,” or both. (*Id.*)

51. In different embodiments, either “all the features are inputted to the classifier” (*Id.* at [0047]) or “selected features are determined,” such that, for example, “the top N features are selected.” (*Id.* at [0049]-[0051]). Sharma discloses that, in one example, “the number of PLS factors to keep” may be “the number of factors which minimize the mean square prediction error.” (*Id.* at [0058].) The “mean square prediction error” is the “mean squared distance between the observed and predicted values.” (*Id.*)

52. Figure 4 shows a method “for **using** a RRF system” “[o]nce the RRF system is trained”—*i.e.*, the classification stage. (*Id.* at [0038].) At that stage, “a computer application is received,” “one or more features are extracted from the

computer application,” and “the PLS and/or Bayesian model classifiers ... are applied to the extracted features to determine a risk-rating of the computer application.” (*Id.*) Sharma’s classification stage does not include any other feature selection.

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

53. I understand that The Petitioner and its expert allege that every limitation of Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, and 52-59 are rendered obvious by Ideses alone, or Ideses in view of Sharma.

Ground	Claims	Argument
1	18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59	§ 103(a): Ideses made the claims obvious.
2	18-20, 22, 24-34, 36, 38-48, 50, 52-59	§ 103(a): Ideses in view of Sharma made the claims obvious

54. 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: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59 Were Not Obvious In View Of Ideses

55. In my opinion, Ideses did not render obvious independent claims 18, 32, or 46. Therefore, it is also my opinion that Ideses did not render any of the dependent claims obvious.

56. As I explain above, Ideses merely discloses “pruning” or selecting features *at the training stage* based on 1) how computationally expensive they are to

extract and evaluate; and 2) empirical determinations of how effective these features are at making a positive categorization of a given application as “desired” or “undesired.”

57. In contrast to Ideses, the ’869 patent discloses selectively turning features on or off at 1) the *classification stage*, 2) based on the values of the static data points that have been extracted from an executable. It is thus my opinion that none of the claims of the ’869 patent are rendered obvious by Ideses.

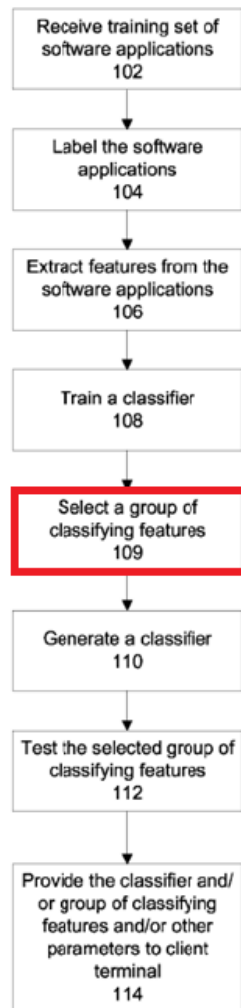
1. Ideses did not render independent claims 18, 32, or 46 obvious

58. As I explain above, the independent claims of the ’869 patent are directed to feature selection at the *classification* rather than the *training* stage, and that selection (i.e. turning features “on” or “off”) is performed based on the extracted content (i.e. the values of the “static data points”) of the executable being evaluated. (EX-1001 7:57-8:22; 12:54-13:4.) In my opinion, Ideses does not disclose the invention recited in Claim 18 because, in Ideses: 1) feature “pruning” occurs while the classifier is being “trained,” and the set of features that will be used to classify an application is determined at the training stage; and 2) the features included in that set will not vary at the classification stage, based on content extracted from the application being classified.

59. In my opinion, the teachings of Ideses relied upon by the Petition are unambiguously directed to performing selection at the “training” stage, based on the

features' effectiveness and computational expense, rather than at the classification stage, based on the content of the executable being classified.

60. Petitioner relies on disclosures that relate only to the training stage. (Petition at 28 citing Ideses at ¶¶ [0071] and [0107]). Both disclosures appear within Ideses's discussion of Figure 1. For example, ¶ [0071] of Ideses starts by declaring that "[r]eference is now made to FIG. 1." And ¶ [0107] of Ideses appears within a discussion of step 109 of Figure 1. These disclosures relate to the training stage because Figure 1 illustrates Ideses's *training stage*, *i.e.*, the steps starting from receiving the training set (step 102) and ending in ultimately "generat[ing] a classifier" (step 110) and "provid[ing] the classifier ... to [the] client terminal" (step 114).



(EX-1005, Fig. 1 (Training), annotated.) The “client terminal” that receives the classifier at the end of Figure 1 is the terminal that later performs the steps of the classification stage, as illustrated in Figure 2. (See EX-2001 ¶¶ 63-67.)

61. Because the feature selection depicted in Figure 1 occurs before Ideses’ system even receives the software application to be classified, *see* EX-1005, Fig. 2, the features chosen by the method depicted in Figure 1 cannot be based on the values of the static data points extracted from the software application to be classified.

62. Similarly, because Ideses only discloses feature “pruning” that has already occurred before an application is even received for classification, that pruning

cannot have any logical nexus to, and thus cannot be “based on,” the values of any static data points contained within, and extracted from, that application.

63. Accordingly, it is my opinion Ideses fails to disclose feature selection at the *classification stage* “based on...a value of one or more static data points,” as required by Claim 1. Moreover, because Petitioner makes no additional argument regarding claims 18, 32 and 46, it is my opinion that Ideses also fails to render obvious any independent claim of the ’869 Patent.

2. Ideses did not render any of the dependent claims obvious

64. Because Petitioner has failed to establish a reasonable likelihood of success on any of the independent claims, it is my opinion that it cannot, and does not, meet its burden for the challenged dependent claims.

B. Ground 2: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59 Were Not Obvious In View Of The Combination of Ideses and Sharma

65. In my opinion, none of the challenged claims of the ’869 patent were rendered obvious by the combination of Ideses and Sharma. Like Ideses, Sharma does not teach feature selection at the classification stage (only at the training stage). Petitioner does not even attempt to argue that Sharma would have made it obvious to modify Ideses to perform feature selection at the classification stage, let alone to do so based on the values of static data points extracted from an executable being evaluated at that stage. Moreover, Petitioner cannot make that argument, because Sharma provides no such teaching. Thus, neither Ideses nor Sharma teaches,

suggests, or renders obvious the idea of turning features on or off during the classification stage.

1. The proposed combination does not remedy the shortcomings of Ideses.

66. Petitioner does not argue that Sharma would have rendered it obvious to introduce classification-stage feature selection into Ideses. Instead, Petitioner argues that, “[t]o the extent ... Ideses does not disclose generating a feature vector [at the classification stage] *using a classifier trained to classify the static data points based on training data*, Sharma explicitly discloses [that feature].” (Petition at 57.) Thus, Petitioner’s use of Sharma is not directed to addressing the shortcoming in Ideses that I identify above—*i.e.*, that feature selection is performed only at the training stage rather than at the classification stage, as claimed.

67. Instead, Petitioner argues that Sharma teaches a different limitation: “generating a feature vector *using a classifier trained to classify the static data points based on training data*.” (*Id.* at 57 (emphasis original).) Even if that were correct—and it is my opinion that it is not correct—generating a feature vector using a trained classifier is not the same as turning features of the feature vector “on or off” based on values of static data points.

68. The portions of Sharma that Petitioner cites confirm that Petitioner’s argument is *not* that Sharma would have remedied the shortcoming of Ideses with respect to turning features on or off at the classification stage. For example,

Petitioner cites Sharma’s disclosure that, when a Bayesian classifier is *trained*, “all the features are inputted to the classifier,” to “determine[] if each feature contributes to the probability that the computer application is malware.” (*Id.* at 58 (citing EX-1006 at [0047]).) Based on this *training* process, “a feature vector is constructed using a Bayesian model classifier.” (*Id.*). That teaching in Sharma plainly describes the *training* stage, which confirms that the Petitioner does not argue that the cited functionality of Sharma discloses turning features “on or off” at the *classification* stage, as required by the claims.

2. Sharma cannot remedy the shortcomings of Ideses.

69. Even had Petitioner wanted to, it *could not* have argued that Sharma teaches turning features on or off at the classification stage.

70. *First*, as I discuss noted above, the cited portion of Sharma (EX-1006 at [0047]) fail to teach that limitation. To start, the cited portion of Sharma discusses the *training* stage, not the *classification* stage. For example, it teaches that, “[t]he probability of each feature **given a computer application is a malware** is computed.” (*Id.* (emphasis added).) In other words, in the analysis being discussed, whether the application being analyzed “is a malware” is “**given**.” The paragraph thus plainly discusses the *training* stage, where the classifier is trained on a set of applications whose characterization—*i.e.*, malicious or benign—is “given.”

71. In fact, the cited portion of Sharma does not teach feature selection at all. Instead, it teaches that “*all* the features are input to the classifier.” (*Id.*) Though the cited portion of Sharma does teach that the classifier “determines if each feature contributes to the probability that the computer application is malware,” it does not teach that any feature is turned “on or off” based on that determination. (*Id.*)

72. And even if the cited Portion of Sharma had taught turning features on or off at the classification stage (it does not), that would still be insufficient because it would not teach that the features are turned on or off *based on the values of static data points*, as claimed.

73. *Second*—beyond the *cited* portions of Sharma—nothing else in Sharma teaches turning features on or off at the classification stage, let alone based on the values of static data points extracted at that stage. While Sharma does teach feature selection (*see, e.g.*, EX-1006 at [0049]), that selection occurs at the training stage, as was typical in the prior art. For example, Sharma teaches that “[f]or benign computer applications, ... “the top N features are selected,” and “[f]or malign computer applications, the same steps as benign computer applications are performed, but those [features] which have already been selected from the benign computer applications are excluded.” (*Id.*) Key to that disclosure is that the features are selected based on the applications’ *pre-known* classifications (*e.g.*, “benign” or “malign”). Thus, this process can occur only at the *training* stage. In contrast, claim 18, recites

turning features on or off for the “executable file” being “evaluat[ed]”—*i.e.*, at the classification stage, where the classification for the file is *not* known. (EX-1001 at 15.) Thus, Sharma does not teach turning features on or off at the classification stage, as claimed.

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

74. 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

75. 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).)

76. 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*.)

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

IX. CONCLUSION

78. 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 20th day of November, 2023 in Irvine, CA.

Respectfully submitted,

A handwritten signature in black ink, appearing to read "Sam Malek", is written over a horizontal line.

Sam Malek, Ph.D.