UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

AO KASPERSKY LAB
Petitioner

v.

WEBROOT, INC.
Patent Owner

IPR2023-_____
U.S. Patent No. 11,409,869

DECLARATION OF VENKATRAMANAN SIVA SUBRAHMANIAN, UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW

TABLE OF CONTENTS

I. INTRODUCTION 5

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE 6

III. LEVEL OF ORDINARY SKILL IN THE ART 16

IV. RELEVANT LEGAL STANDARDS 17

A. Claim Construction 18

B. Anticipation Under 35 U.S.C. § 102 19

C. Obviousness Under 35 U.S.C. § 103 20

V. OVERVIEW OF THE '869 PATENT 23

A. Summary of the '869 Patent 23

B. Overview of Prosecution History 25

VI. Technical Background Known to a Person of Ordinary Skill in the Art 28

A. Character strings 28

B. Executable encryption/packing, and decryption/unpacking 29

C. Static and dynamic malware analysis 31

D. Static data points 32

E. Machine learning models including support vector machines 33

VII. OVERVIEW OF THE PRIOR ART 40

A. OVERVIEW U.S. PATENT APPLICATION PUBLICATION NO. 2015/0213376 TO IDESES ("IDESES") 40

B. OVERVIEW OF U.S. PATENT APPLICATION PUBLICATION NO. 2016/0154960 TO SHARMA ("SHARMA") 46

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE 48

A. Ground 1: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59 are obvious under 35 U.S.C. § 103 by Ideses 48

1. Claim 18 48

2. Claim 20 77

3. Claim 22 79

4. Claim 24 82
5. Claim 25 84
6. Claim 26 85
7. Claim 27 87
8. Claim 28 88
9. Claim 29 89
10. Claim 30 90
11. Claim 31 93
12. Claim 32 95
13. Claim 34 99
14. Claim 36 100
15. Claim 38 101
16. Claim 39 102
17. Claim 40 103
18. Claim 41 104
19. Claim 42 104
20. Claim 43 105
21. Claim 44 106
22. Claim 45 107
23. Claim 46 108
24. Claim 48 112
25. Claim 50 113
26. Claim 52 114
27. Claim 53 115
28. Claim 54 116
29. Claim 55 117
30. Claim 56 117
31. Claim 57 118
32. Claim 58 119

33. Claim 59 120

B. Ground 2: Claims 18-20, 22, 24-34, 36, 38-48, 50, 52-59 are obvious under 35 U.S.C. § 103 over Ideses in view of Sharma 121

1. Reasons to Combine Ideses and Sharma 121

2. Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59 134

3. Elements [18.5.b], [32.5.b], and [46.5.b] (as recited above in Section VIII.A.) 134

4. Claim 19 137

5. Claim 33 142

6. Claim 47 143

IX. NO OBJECTIVE INDICIA OF NONOBVIOUSNESS 144

X. CONCLUSION 144

I, Venkatramanan Siva Subrahmanian, do hereby state and declare as follows:

I. INTRODUCTION

1. I have been retained by DeWitt LLP, counsel for Petitioner, AO Kaspersky Lab in the matter of the *Inter Partes* Review of U.S. Patent No. 11,409,869 ("the '869 patent") to Schmidtler et al. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

2. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this investigation. My compensation is not contingent on the outcome of this matter or the specifics of my testimony, and I have no other interest in this case or the parties thereto.

3. I have been asked to provide my opinions regarding whether claims 18-20, 22, 24-34, 36, 38-48, 50, 52-59 (the "Challenged Claims") of the '869 patent are unpatentable as they would have been anticipated or obvious to a person of ordinary skill in the art ("POSITA") at the time of the alleged invention, in light of prior art. It is my opinion that all limitations for the Challenged Claims would have been anticipated or obvious to a POSITA in light of the prior art. My opinions are based on my experience, knowledge and skill in computer security and cyber-security technology.

4. In preparation for this declaration, I have reviewed '869 patent, its prosecution history, and all of the exhibits cited in this declaration and Petition. I may rely upon these materials, my experience and knowledge, and/or additional materials to rebut arguments raised by the Patent Owner. Further, I may also consider additional documents and information in forming any necessary opinions, including documents that may not yet have been provided to me.

5. I am over the age of 18 and make this Declaration based upon personal knowledge, experience, and investigation that I have conducted. I will provide testimony concerning my statements, opinions and conclusions set forth below and may amplify or add thereto through such testimony, demonstratives, or in further written declarations.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

6. I am the Walter P. Murphy Professor of Computer Science with tenure in the McCormick School of Engineering, Northwestern University (September 2021-present) and Faculty Fellow in the Buffett Institute of Global Affairs. Prior to this appointment, I was the Dartmouth College Distinguished Professor in Cybersecurity, Technology, and Society at Dartmouth College with tenure in the Computer Science Department (August 2017-August 2021) and Director of the Institute for Security, Technology and Society (April 2018-August 2021). Prior to

that appointment, I was on the faculty of the University of Maryland's Computer Science Department from 1989 to 2017, as an Assistant Professor from 1989 to 1995, as an Associate Professor with tenure from 1995 to 2000 and as a full Professor from 2000 to 2017. I also served as Director of the University of Maryland's Institute for Advanced Computer Studies.

7. I graduated from Syracuse University with a Master of Science degree in Computer Science (1987) and Doctor of Philosophy degree in Computer Science (1989).

8. I have particular expertise in cyber-security having developed some of the first secure query answering algorithms, flexible authentication frameworks, unexplained behavior detection algorithms, techniques for the scalable detection of known threats, identifying human behaviors linked with cyber-vulnerabilities, and combating intellectual property theft. I led the development of the first ever Global Cyber Vulnerability Report consisting of the analysis of billions of anti-virus telemetry reports to quantify the vulnerability of 44 major nations. I developed some of the first methods to predict which vulnerabilities will be exploited, when, and how severe they will be. In addition, I have developed methods for malware identification, deploying deception within networks in order to deceive attackers, and separating false security alarms from real ones. I also have particular expertise in machine learning. I have taught machine learning several times at three

educational institutions (University of Maryland College Park, Dartmouth College, and Northwestern University) since Spring 2014. In addition, I have written numerous articles, given numerous invited talks and/or keynotes around the world based on machine learning models for a variety of security problems.

9. As evidence of my expertise and leadership in computer security, cyber-security, and machine learning, I have written numerous papers, edited several important books, and been invited to speak at many international conferences on the subject matter. See my attached CV for a full listing of my writings and presentations, which is attached to this report as Exhibit 1003. A sample of my recent books and keynotes include:

A. Author of *The Global Cyber-Vulnerability Report* (Springer 2013) with M. Ovelgonne, T. Dumitras, and B. A. Prakash.

B. S. Jajodia, G. Cybenko, V.S. Subrahmanian, V. Swarup, C. Wang, and M. Wellman. (eds). Adaptive, Autonomous, Secure, Cyber Systems, Springer, 2020.

C. S. Jajodia, V.S. Subrahmanian, V. Swarup, C. Wang (eds). Cyber-Deception: Building the Scientific Foundation, Springer, 2016.

D. S. Jajodia, P. Shakarian, V.S. Subrahmanian, V. Swarup, C. Wang (eds). Cyber-Warfare: Building the Scientific Foundation, Springer 2015. *Also released as Chinese student edition in 2016.*

E. Moving Target Defense II (eds. S. Jajodia, A.K. Ghosh, Subrahmanian, V.S., Swarup, V., Wang, C., Wang, X.S.), Springer, September 2012. *Also released in 2014 as a special Chinese edition.*

F. E. Lancaster, T. Chakraborty and V.S. Subrahmanian. MALT^{P}: Parallel Prediction of Malicious Tweets, *IEEE Transactions on Computational Social Systems*, Vol 5, Nr. 4, pages 1096-1108, Dec 2018.

G. Invited Tutorial, 2019 Intl. Summer School on Machine Learning & Cybersecurity, Padua, Italy Sep 2019.

H. N. Park, E. Serra, and V.S. Subrahmanian. Saving Rhinos with Predictive Analytics, *IEEE Intelligent Systems,* Vol. 30, Nr. 4, pages 86-88, July/August 2015.

I. Keynote Speaker, 8th International Workshop on Parallel and Distributed Computing for Large-Scale Machine Learning and Big Data Analytics (ParLearning 2019), in conjunction with the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2019), Anchorage, Alaska, Aug 2019.

J. V.S. Subrahmanian, C. Pulice, J. Brown and J. Bonen-Clark. A Machine Learning Model of Boko Haram, Springer, Jan 2021. *Foreword by Geert Kuiper, former Deputy Director of the Netherlands Defence Intelligence and Security Service.*

K. Keynote Speaker, 1st International Conference on Autonomous Intelligent Cyber-defence Agents, Paris France (but via Zoom), March 2021.

L. Keynote Speaker on Ransomware, United Health Group Healthcare Technology Advisory Council, June 30, 2021. (Via Zoom)

M. Keynote Speaker, 18th International Conference on Security and Cryptography (SECRYPT 2021), July 2021. (Via Zoom)

N. Keynote Address on AI & Cybersecurity, Wells Fargo *Confluence* conference, Dec 2021. (via Zoom)

O. Invited Talk, DARPA SocialCyber AI Exploration PI meeting, June 2022.

P. Keynote Address on AI & Cybersecurity, Italian Cybersecurity Conference, May 2023, Bari Italy.

10. Further details of my qualifications are presented in my curriculum vitae, which is attached to this report as Exhibit 1003. By my education, training, and experience, as evidenced by my curriculum vitae, I am qualified as one skilled in the art of computer security and cyber-security technology.

11. I have considered information from various sources in forming my opinions. Besides drawing from my experience in the computer industry, I also have reviewed

the ’591 patent and its file history, the other documents and references as cited herein, and general technical references. The exhibits are:

A. Ex. 1001: U.S. Patent No. 11,409,869 (issued Aug. 9, 2022).

B. Ex. 1002: Declaration of V.S. Subrahmanian, Ph.D. under 37 C.F.R. §1.68.

C. Ex. 1003: Curriculum Vitae of V.S. Subrahmanian, Ph.D.

D. Ex. 1004: File Hist. U.S. Patent No. 11,409,869

E. Ex. 1005: U.S. Patent Publication No. 2015/0213376 to Ideses et al. (published Jul. 30, 2015) (“Ideses”).

F. Ex. 1006: U.S. Patent Application Publication No. 2016/0154960 to Sharma et al. (published Jun. 2, 2016) (“Sharma”).

G. Ex. 1007: CHARLES LEDOUX & ARUN LAKHOTIA, *Malware and Machine Learning*, *in* INTELLIGENT METHODS CYBER WARFARE 1 (2014).

H. Ex. 1008: U.S. Patent Application Publication No. 2015/0213365 to Ideses et al. (“Ideses ‘365”).

I. Ex. 1009: Claim Construction Order, Mar. 16, 2023, Case No. 6:22-CV-00243-ADA-DTG.

J. Ex. 1010: Joint Stipulation Reserving Appellate Rights, Apr. 12, 2023, Case No. 6:22-CV-00243-ADA-DTG.

K. Ex. 1011: Kris Kendall, *Practical Malware Analysis*, *in* BLACK HAT CONF., USA (August 2007).

L. Ex. 1012: Hao Helen Zhang et al., *Compactly Supported Radial Basis Function Kernels*, INST. STAT. MIMEO SERIES NO. 2570, N. CAROLINA ST. U. DEP'T. STAT. 2 (2004).

M. Ex. 1013: Zane Markel & Michael Bilzor, *Building a Machine Learning Classifier for Malware Detection*, 2014 SECOND WORKSHOP ANTI-MALWARE TESTING RES. (WATER) (2014).

N. Ex. 1014: Naser Peiravian & Xingquan Zhu, *Machine Learning for Android Malware Detection Using Permission and API Calls*, *in* 2013 IEEE 25TH INT'L CONF. TOOLS ARTIFICIAL INTELLIGENCE 300 (2013).

O. Ex. 1015: Ekta Gandotra et al., *Malware Analysis and Classification: A Survey*, J. INF. SECURITY 5, 56-64 (2014).

P. Ex. 1016: Rafiqul Islam et al., *Classification of Malware Based on Integrated Static and Dynamic Features*, 36 J. NETWORK COMPUTER APPLICATIONS, 646 (2012).

Q. Ex. 1017: D. Michael Cai et al., *Comparison of feature selection and classification algorithms in identifying malicious executables*, 51(6) COMPUTATIONAL STAT. DATA ANALYSIS 3156 (2007).

R. Ex. 1018: Guanhua Yan et al., *Exploring Discriminatory Features for Automated Malware Classification*, 10 DETECTION INTRUSIONS MALWARE, VULNERABILITY ASSESSMENT: 10TH INT'L CONF., 41 (2013).

S. Ex. 1019: Rafiqul Islam et al., *Classification of Malware Based on String and Function Feature Selection*, 2010 SECOND CYBERCRIME TRUSTWORTHY COMPUTING WORKSHOP 9 (2010).

T. Ex. 1020: Eitan Menahem et al., *Improving Malware Detection by Applying Multi-Inducer Ensemble*, COMPUTATIONAL STAT. DATA ANALYSIS, 53(4), 1483 (2008).

U. Ex. 1021: Ianir Ideses & Assaf Neuberger, *Adware Detection and Privacy Control in Mobile Devices*, *in* 2014 IEEE 28TH CONVENTION ELECTRICAL ELECTRONICS ENGINEERS ISRAEL 1 (2014).

V. Ex. 1022: Raja Khurram Shahzad et al., 2011, *Accurate Adware Detection Using Opcode Sequence Extraction*, *in* 2011 SIXTH INT'L CONF. AVAILABILITY, RELIABILITY SECURITY 189 (2011).

W. Ex. 1023: Raymond Canzanese et al., *Toward an Automatic, Online Behavioral Malware Classification System*, *in* 2013 IEEE 7TH INT'L CONF. SELF-ADAPTIVE SELF-ORGANIZING SYS., 111 (2013).

X. Ex. 1024: Yanfang Ye et al., *Automatic Malware Categorization Using Cluster Ensemble*, *in* PROC. 16TH ACM SIGKDD INT'L CONF. KNOWLEDGE DISCOVERY DATA MINING 95 (2010).

Y. Ex. 1025: Asaf Shabtai et al., *Detection of Malicious Code by Applying Machine Learning Classifiers on Static Features: A State-Of-The-Art Survey*, INFO. SECURITY TECH. REP. 14(1), 16 (2009).

Z. Ex. 1026: A.M. Aswini & P. Vinod, *Droid Permission Miner: Mining Prominent Permissions for Android Malware Analysis*, *in* FIFTH INT'L CONF. APPLICATIONS DIGITAL INF. WEB TECH. 81 (2014).

AA. Ex. 1027: Konrad Rieck et al., *Learning and Classification of Malware Behavior*, *in* DETECTION INTRUSIONS MALWARE, VULNERABILITY ASSESSMENT: 5TH INT'L CONF., 108 (2008).

BB. Ex. 1028: Veelasha Moonsamy et al., *Feature Reduction to Speed Up Malware Classification*, *in* INF. SECURITY TECH. APPLICATIONS: 16TH NORDIC CONF. SECURE IT SYS., NORDSEC 2011, TALLINN, EST., OCT. 26-28, 2011, REV. SELECTED PAPERS 16, 176 (2012).

CC. Ex. 1029: John P. Dickerson et al., *Using Sentiment to Detect Bots on Twitter: Are Humans More Opinionated Than Bots?*, *in* 2014 IEEE/ACM INT'L CONF. ADVANCES SOCIAL NETWORKS ANALYSIS MINING 620 (2014).

DD. Ex. 1030: Wei Yu et al., *Towards Neural Network Based Malware Detection on Android Mobile Devices*, CYBERSECURITY SYS. HUMAN COGNITION AUGMENTATION 99 (2014).

EE. Ex. 1031: Guanhua Yan et al., 2013. *Exploring Discriminatory Features for Automated Malware Classification*, *in* DETECTION INTRUSIONS MALWARE, VULNERABILITY ASSESSMENT: 10TH INT'L CONF., DIMVA 2013, BERLIN, GER., JULY 18-19, 2013 PROC. 10, 41 (2013).

FF. Ex. 1032: Julia Yu-Cheng et al., *An Information Retrieval Approach for Malware Classification Based on Windows API Calls*, *in* 4 PROC. 2013 INT'L CONF. MACHINE LEARNING CYBERNETICS 1678 (2013).

GG. Ex. 1033: Zi Chu et al., *Detecting Automation of Twitter Accounts: Are You a Human, Bot, or Cyborg?*, 9 IEEE TRANSACTIONS DEPENDABLE SECURE COMPUTING 6, 811 (2012).

HH. Ex. 1034: Axelle Apvrille, *Android Reverse Engineering Tools From an Anti-Virus Analyst's Perspective*, slides 11-15 (March 2012), http://wikisec.free.fr/papers/insomnidroid.pdf.

II. Ex. 1035: Li Sun et al., *Pattern Recognition Techniques for the Classification of Malware Packers*, *in* 15 PROC. INF. SECURITY PRIV.:

15TH AUSTRALASIAN CONF., ACISP 2010, SYDNEY, AUSTL., JULY 5-7, 2010, 370 (2010).

JJ. Ex. 1036: Munkhbayar Bat-Erdene et al., *Dynamic Classification of Packing Algorithms for Inspecting Executables Using Entropy Analysis*, *in* 2013 8TH INT'L CONF. MALICIOUS UNWANTED SOFTWARE: "AMERICAS" (MALWARE) 19 (2013).

KK. Ex. 1037: M. Zubair et al., *PE-Probe: Leveraging Packer Detection and Structural Information to Detect Malicious Portable Executables*, *in* 8 PROC. VIRUS BULL. CONF. (2009).

III. LEVEL OF ORDINARY SKILL IN THE ART

12. I understand the level of a person of ordinary skill in the art at the time of the claimed invention is determined through consideration of several factors. I understand that those factors include: (1) the educational level of the inventor(s); (2) type of problems encountered in the art; (3) prior art solutions to those problems; (4) rapidity with which innovations are made; (5) sophistication of the technology; and (6) educational level of active workers in the field.

13. After considering the factors mentioned above, and based on the disclosure of the '869 Patent, it is my opinion that a person of ordinary skill in the art at the relevant time for these patents would hold a Bachelor's degree in computer science

or a similar field, have about 2 years of experience as a software programmer or similar title, and have about 2 years of experience specifically in the computer security field. A person having more education, such as a master's degree, would require less experience, and a person not having at least a bachelor's degree would need more experience.

IV. RELEVANT LEGAL STANDARDS

14. In forming my opinions and considering the subject matter of the '869 patent and its claims in light of the prior art, I am relying on certain legal principles that counsel in this case explained to me. My understanding of these concepts is summarized below.

15. I understand that the claims define the invention. I also understand that an unpatentability analysis is a two-step process. First, the claims of the patent are construed to determine their meaning and scope. Second, after the claims are construed, the content of the prior art is compared to the construed claims.

16. I understand that a claimed invention is only patentable when it is new, useful, and non-obvious in light of the "prior art." That is, the invention, as defined by the claims of the patent, must not be anticipated by or rendered obvious by the prior art.

A. Claim Construction

17. I understand that patent claims are generally interpreted according to their ordinary and customary meaning to a person of ordinary skill in the art taking into consideration the so-called "intrinsic evidence" of the patent consisting of (1) the claim language; (2) the specification and drawings; and (3) the prosecution history. I understand that a Court has discretion to take into consideration so-called "extrinsic evidence" including references (prior art and non-prior art) as well as definitions from dictionaries and treatises to the extent they do not contradict intrinsic evidence.

18. I understand that claim terms may be explicitly defined in the patent specification, or they may be implicitly defined through consistent usage in the specification. I also understand that the scope of claim terms may be limited by statements in the specification or prosecution history where the application clearly disavows or disclaims subject matter in a clear and unmistakable manner.

19. I understand that a patent is required to provide a written description of the invention, and the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, to make and use the claimed subject matter.

20. I understand that a patent claim is indefinite if, when read in light of the specification and the prosecution history, the claim fails to inform, with reasonable certainty, those skilled in the art about the scope of the invention.

21. It is my understanding that, although the terms of a claim may appear to be definite, inconsistency with the specification disclosure or prior art teachings may make an otherwise definite claim take on an unreasonable degree of uncertainty. It is also my understanding that indefiniteness may result from inconsistent prosecution history statements where the claim language and specification on their own leave an uncertainty that, if unresolved, would produce indefiniteness.

22. It is also my understanding that a fundamental rule of claim construction is that terms in a patent are construed with the meaning with which they are presented in the patent, and that if the specification reveals a special definition given to a claim term by the patentee that differs from the meaning it would otherwise possess, then the patentee's definition governs.

B. Anticipation Under 35 U.S.C. § 102

23. I understand that a patent claim is invalid as anticipated if a single piece of prior art teaches every element of the claims, viewed from the perspective of a person of ordinary skill in the art. I also understand that an anticipatory reference does not have to recite word for word what is in the anticipated claims. Anticipation can also

occur when a claimed limitation is inherent in the relevant reference. I have been advised that if the prior art necessarily functions in accordance with, or includes, the claims limitations, it can anticipate even though the limitation is not expressly disclosed.

C. Obviousness Under 35 U.S.C. § 103

24. I understand that a patent claim is invalid as obvious if the claimed invention would have been obvious to a person of ordinary skill in the art at the time the claimed invention was made. This means that even if all of the elements of the claim cannot be found in a single prior art reference that would anticipate the claim, a person of ordinary skill in the field who knew about all the prior art would have come up with the claimed invention. I understand that in an obviousness determination, the person of ordinary skill in the art is presumed to have knowledge of all material prior art. I understand that whether a claim is obvious is based upon the determination of several factual issues. A patent is invalid as obvious when a POSITA (1) would have been motivated to combine the teachings of the prior art to achieve and the claimed invention and (2) would have had a reasonable expectation of successfully producing the claimed invention through the combination.

25. In considering obviousness, I understand that one must determine the scope and content of the prior art. I understand that, in order to be considered as prior art

to a patent being considered, a prior art reference must be reasonably related to the claimed invention of that patent. A reference is reasonably related if it is in the same field as the claimed invention or is from another field to which a person of ordinary skill in the art would look to solve a known problem.

26. I understand that one must determine what differences, if any, existed between the claimed invention and the prior art.

27. I understand that a patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, one may consider whether a reason has been identified that would have prompted a person of ordinary skill in the art to combine the elements or concepts from the prior art in the same way as in the claimed invention. There is no single way to define the line between true inventiveness on the one hand (which is patentable) and the application of common sense and ordinary skill to solve a problem on the other hand (which is not patentable). For example, market forces or other design incentives may be what produced a change, rather than true inventiveness.

28. I understand that one may consider whether (1) the change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness; (2) there is some teaching or suggestion in the prior art to make the modification or combination of elements

claimed in the patent; (3) the innovation applies a known technique that had been used to improve a similar device or method in a similar way; (4) the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those skilled in the art; (5) the invention merely substituted one known element for another known element in order to obtain predictable results; (6) the invention merely applies a known technique to a known device, method, or product to yield predictable results, or (7) known work in the field may have prompted variations of use of the same inventions in the same or different fields due to market forces or design incentives that would have been predictable to a person of ordinary skill in the art.

29. I understand that one must be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact. I understand that the determination should be based on the position of a person of ordinary skill in the field at the time the claimed invention was made and should not consider what is known today or what is learned from the teaching of the patent.

30. I understand that so-called "secondary considerations of non-obviousness," must be considered in an obviousness analysis. I understand that an analysis cannot use hindsight in determining whether a patent claim is obvious. I understand that secondary considerations of non-obviousness include: (a) a long-felt but unresolved

need for the invention; (b) commercial success of the invention; (c) copying of the invention; (d) praise and recognition of the invention by others; (e) licensing of the rights to the invention; and (f) unexpected results.

V. OVERVIEW OF THE '869 PATENT

31. The '869 patent is entitled "Automatic Threat Detection of Executable Files Based on Static Data Analysis." Ex. 1001, Cover. I understand that the '869 patent was filed on February 14, 2020, as application number 16,791,649 and claims priority to application number 14/709,874 (patent number 10,599,844), filed on May 12, 2015. For the purpose of my analysis, I have assumed the effective filing date is the filing date of parent application 14/709,874, May 12, 2015.

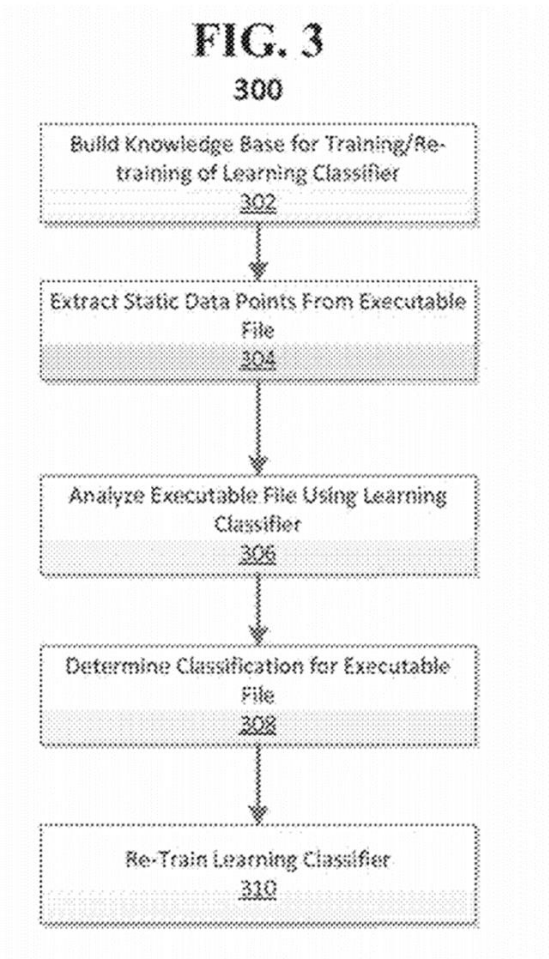
A. Summary of the '869 Patent

32. The '869 patent relates to training a classifier to determine whether an executable file is one of three classification types, harmful, benign, or potentially unwanted, using static data points (e.g., file size, code, size, product name) "before the executable is actually executed on a processing device." Ex. 1001, Abstract, 1:31-32, 2:12-14, 4:43-46, Table 1.1.

33. The '869 patent discloses devices, methods, and systems for training a classifier and extracting static data points from an executable file, analyzing the

executable file with the classifier, and determining the file's classification type (e.g., potentially unwanted). Ex. 1001, Abstract, 1:31-45. The classifier "is trained from data comprising known malicious executable files, known benign executable files and known unwanted executable files." Ex. 1001, Abstract, 1:38-40.

34. Figure 3 of the '869 patent depicts an "exemplary method" for performing threat detection:



Ex. 1001, FIG. 3. At operation 302, a knowledge base is built using known malicious executable files, known benign executable files, and known potentially unwanted executable files and used to train/re-train a classifier to evaluate threats in executable files. Ex.1001, 11:60-12:4. At operation 304, when an executable file is identified for evaluation, static data points are extracted from an executable file. Ex. 1001, 12:11-13. At operation 306, the executable file is analyzed for threats. Ex. 1001, 12:43-44. At operation 308, a classification for the executable file is determined. Ex. 1001, 13:12-13.

B. Overview of Prosecution History

35. It is my understanding that the '869 patent is a U.S. Application filed February 14, 2020, as a continuation of U.S. Patent No. 10,599,844 ("the '844 patent"), filed May 12, 2015. A determination of whether the '869 patent is entitled to the earliest alleged priority date is unnecessary as the prior art relied upon pre-dates the earliest alleged priority date.

36. It is my understanding that during the prosecution of the '869 patent there were a number of office actions rejecting the claims and a number of amendments were made to the claims.

37. It is my understanding that a first Non-Final Rejection was mailed March 29, 2021, in which all independent claims were rejected but allowable subject matter was indicated for several dependent claims.

38. It is my understanding that, in response to the first Non-Final Rejection, Applicant amended the independent claims to include the requirement that "wherein one or more features of the feature vector are selectively turned on or off based on information associated with one or more of the plurality of extracted static data points," in an Amendment filed June 17, 2021. Ex. 1004, 213. It is further my understanding that this amended language did not fully copy the language from the claims deemed to include allowable subject matter from the first Non-Final Rejection. It is further my understanding that Applicant also filed a Terminal Disclaimer to overcome a double patenting rejection over the parent '844 patent. Ex. 1004.

39. It is my understanding that a first Notice of Allowance was mailed August 11, 2021, in response to the Applicant's amendments and arguments. Ex. 1004, 181.

40. It is my understanding that in response to the first Notice of Allowance, on November 18, 2021, Applicant filed a request that prosecution be continued, and additional amendments and new claims be entered. Ex. 1004, 157-169.

41. It is my understanding that in response to Applicant's request for continued examination and amendments, a Non-Final Rejection was mailed December 21, 2021, in which allowable subject matter was indicated. Ex. 1004, 132.

42. It is my understanding that in response to the Non-Final Rejection, the Applicant further amended the independent claims to recite, in pertinent part, "wherein one or more features of the feature vector are selectively turned on or off ~~based on information associated with one or more of the plurality of extracted static data points~~ <u>based at least in part on evaluation of whether a value of one or the plurality of static data points is within a predetermined range</u>" and cancelled dependent claims reciting the same in an Amendment filed February 8, 2022. Ex. 1004, 123. It is further my understanding that the amended claim language for the independent claims tracks with the indicated allowable subject matter from the December 21, 2021, Non-Final Office Action.

43. It is my understanding that in response to Applicant's amendment on February 8, 2022, a second Notice of Allowance was mailed April 15, 2022. Ex. 1004, 75.

44. It is my understanding that in response to the second Notice of Allowance, on May 10, 2022, Applicant filed a request that prosecution be continued, and new claims be entered. Ex. 1004, 54.

45. It is my understanding that in response to Applicant's request for continued prosecution, the Examiner concluded that the application was patentable over the

prior art and issued a third Notice of allowance. It is further my understanding that the Examiner indicated that the application was patentable over the prior for the following reason: "the prior art, either alone or in a reasonable combination, fails to teach selectively turning on/off one or more features of the generated feature vector based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range," amendments that were made to the independent claims (or initially added to new independent claims). Ex. 1004, 23.

46. It is my understanding that the '869 patent issued on August 9, 2022, with the claims as indicated in the amendments filed May 10, 2022. Ex. 100, Cover.

VI. Technical Background Known to a Person of Ordinary Skill in the Art

A. Character strings

47. A character string is a commonly used data structure and data type. A character is typically any symbol present on a computer keyboard. A character string is a sequence of characters of the kind that are typically found on a computer keyboard. For instance, "#10DowningStreet" is an example of a character string. Character strings can be used for many different purposes on a computer.

48. A "predefined character string" is a character string that is typically defined in advance such that the predefined character string is a predetermined specific

sequence of characters. For example, when analyzing a software application, one might look for pre-defined strings that allow the application to connect to an external server or a website or allow the application to call a phone (e.g., a premium phone number where the caller incurs a fee for calling the premium phone line). In the latter case, the pre-defined strings would be strings representing a list of predefined and predetermined premium phone numbers.

B. Executable encryption/packing, and decryption/unpacking

49. An executable is a program/file represented in its low-level machine readable format. Computer programs are typically written in a high-level language that human programmers can understand. Java, C, and C++ are some well-known high-level programming languages. However, for a computer to be able to execute these high-level programs, they must be represented in a low-level computer readable format. This is usually done by running the high-level program through a program called a "compiler" that converts the high-level program into a low-level, machine readable program that can be executed. These low-level programs are often called executables.

50. Encryption is the process of converting information/data into a code generally to obscure the information's meaning. Decryption is the process of reversing the encryption of information/data. Encryption algorithms take a computational object

as input (e.g., an executable file) and convert them into an encrypted form. Once encrypted, it is usually not possible for normal users to understand the material in the encrypted file. Decryption algorithms convert an encrypted file back into its original form.

51. Packing is the process of encrypting and compressing data. Unpacking is reversing the packing process. According to LeDoux and Lakhotia in 2014, "Packing is a process whereby an arbitrary executable is taken and encrypted and compressed into a 'packed' form that must be uncompressed and decrypted, i.e., 'unpacked', before execution. This packed version of the executable is then packaged as data inside another executable that will decompress, decrypt, and run the original code. Thus, the end result is a new binary that looks very different from the original, but when executed performs the exact same task, albeit with some additional unpacking work. A program that does packing is referred to a packer and the newly created executable is called the packed executable." Ex. 1007, 11.

52. A slightly different definition of packed programs was provided by Kendall and McMillan in a 2007 BlackHat conference paper. They state, "[o]ne of the major complicating factors in performing malware analysis is the proliferation of programs that modify an executable file to obfuscate its contents and hide the actual program logic from a reverse engineer performing static analysis. Programs that modify other program files to compress or disguise their contents are most commonly referred to

as ‘executable packers’ or just ‘packers.’ When a packer compresses, encrypts, or otherwise modifies an executable program, the program looks much different from the static analysis perspective, but still runs as it did before it was ‘packed.’ Once a program has been packed, the original program’s logic and other metadata are very hard to recover through static analysis.” Ex. 1011, 5.

53. Both of the above definitions make clear that a person of skill in the art would have understood the term “packer” or “packing” to cover both compression and encryption.

C. Static and dynamic malware analysis

54. Static analysis typically refers to the process of examining code as it stands. For instance, if the code is written in Java, static analysis may analyze the size of the code (in Kilobytes), the number of lines of code, the number of different calls made within the code to external Internet domains, any phone numbers present in the code, whether parts of the code are encrypted, and so forth. For static analysis, the code is not run or executed. Accordingly, static analysis includes determination of characteristics of a program that can be obtained without running or executing the program code.

55. In contrast, dynamic analysis analyzes the behavior of the code when it is executed.

56. As stated by Kendall and McMillan in their 2007 BlackHat paper: "[w]hen performing static analysis, you are in essence conducting an autopsy of the code—examining it at rest, in a dead state. When performing dynamic analysis, you actually run the malware and observe its actions. Is [sic] essence, you will create a fishbowl for the malware and then watch what it does." Ex. 1011, p. 7.

D. Static data points

57. In the context of the '869 patent, static data points are discrete points of data within a file that are obtained through static analysis, rather than dynamic analysis (as described above). It is stated in the '869 patent that static data points can be extracted from decrypted and unpacked files and that static data points can be extracted from encrypted and packed files without the need to decrypt or unpack the file. Ex. 1001, Abstract; 6:44-47; 12:17-21. A person of ordinary skill at the priority date of the '869 patent would have understood a static data point to be a static feature (i.e., a feature associated with a software application that can be obtained through a static analysis of the software application's code). Static analysis, as described herein, includes determination of characteristics (i.e., features) of a program that can be obtained without running or executing the program code.

58. Table 1.1 of the '869 patent lists a number of static data points that can be extracted from a file whether decrypted, unpacked, packed (without unpacking the

file), or encrypted (without decrypting the file). Ex. 1001, Table 1.1. According to the '869 patent, the static data points listed in Table 1.1 fall into 4 categories: numeric values (e.g., file size), nominal values (e.g., file OS), strings/byte sequences (e.g., company name), and boolean values (e.g., certificate validity). Examples further provided in the '869 patent of static data points "include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples." Ex. 1001, 5:62-67.

E. Machine learning models including support vector machines

59. I have taught machine learning at three institutions over time: University of Maryland, Dartmouth College, and Northwestern University. In every edition of my machine learning course, I have taught Support Vector machines. I have written numerous papers where machine learning and/or support vector machines have been used. The description below is based on my personal expertise with both machine learning and support vector machines.

60. A machine learning model is a program that can find patterns, make predictions, or make decisions based on previously unseen data sets. At the time of the priority date of the '869 patent, machine learning models were widely used for

predicting classification of data. In other words, the model tried to predict whether an object (e.g., a person or a computer application) belonged to a certain class (e.g., being at risk of diabetes in the case of a person or being malware in the case of an application).

61. To develop a machine learning model, several steps are required. The developer may first define a set of features for the type of object being studied (e.g., person or application) and the phenomenon being predicted. The list of values of such features is usually called a feature vector. For instance, when trying to predict whether a person will be at risk of diabetes, the list of considered features may include the body mass index (or BMI) of the person and the average amount of exercise the person performs per day in minutes, as well as many other features. In the case of predicting whether a computer application is benign or malware, the list of features may include the size of the application, whether it is signed or not, whether it seeks a certain permission, and so forth. A feature vector is an ordered list of feature values. In the case of the diabetes example with the BMI and exercise features (25,60) may be a feature vector associated with a person whose BMI is 25 and who exercises for 60 minutes per day on average. When the values of features (e.g., the size of an executable app) can be extracted automatically, the process of extracting feature values is called feature extraction.

62. Another step in developing a machine learning model (which may occur before, after, or concurrently with the first step), is to collect a dataset (e.g., of people whose risk of diabetes is known or applications whose status as benign or malicious is known). This dataset is called a "training set."

63. In a next step, each object in the training set (e.g., person or app) is characterized based on the values of the features defined previously for the machine learning model. The list of feature values associated with an object (e.g., person or app) is called a feature vector. Table 1 below shows a tiny sample training set with 7 people. The feature vector corresponding to Jim is (35,5) and Jim (and his associated feature vector belongs to class 1 (diabetic). In contrast, the feature vector associated with Mary is (20,60) and her feature vector belongs to class 0 (non-diabetic). Sometimes, these (feature vector, class) pairs are also called the "training set."

Person	BMI	Exercise per Day	Class
Jim	35	5	1
Mary	20	60	0
John	25	45	0
Lisa	30	15	1
Ed	25	35	0
Donna	22	40	0
Susan	40	25	1

Table 1

In the training set, we know the correct class to which the object (or analogously its associated feature vector) belongs.

64. Finally, a machine learning algorithm (also called a classifier) is used to learn a “model” or “predictive model” from the training set. This step is also referred to as “training a classifier.” Such a model (once learned or trained) can automatically identify a condition on the features such that when an object’s feature vector satisfies the condition, it is very likely that the object belongs to one class (e.g. person it at risk of diabetes, or application is malware) and when the object’s features do not satisfy the condition, the object very likely belongs to the other class (e.g. person is not at risk of diabetes, or application is benign).

65. Such machine learning models are called “binary” classifiers because they try to take an object’s associated “feature vector” (i.e., list of feature values) as an input and classify it into one of two classes. There are many different types of binary classifiers. Even though the discussion above tries to separate objects’ feature vectors into two classes, there are many easy and well-known methods to take such binary classifiers and adapt them to classify objects into more than two classes.

66. A Support Vector Machine (or SVM) is a certain type of machine learning model (i.e., binary classifier). For instance, suppose we consider the training set about people shown in Table 1 and we want to predict who is at risk of diabetes. The training set we have in this specific example can then be represented as a 2-dimensional feature space (2-dimensional because we have only two features). Figure 1 below shows this training set represented as a 2-dimensional feature space.

The 2-dimensional space shown in Figure 1 (i.e., the entire grid) is called the "feature space" in this case. Each point in the feature space corresponds to an object (in this case, a person) in the training set. For instance, the point for the feature vector (40,25) corresponds to Susan. This feature vector is shown in the feature space as red because Susan is a known diabetic. In contrast, the point for the feature vector (20,60) corresponds to Mary and the point (20,60) is shown in the feature space as green because the training set says Mary does not have diabetes.

67. A support vector machine learns to predict the classification of a "new" set of data by learning a line in the feature space which mostly separates the training data into their respective classifications (e.g., a line through the feature space that separates those in the training set with diabetes from those in the training set without). One such line is shown in orange in Figure 1. We see that all points on one side of the line are green and all points on the other side of the line are red. Therefore, this line achieves a perfect separation and is called a "separator" line. The equation of this line is y-x=0. The line splits the feature space into two parts: the part "above" the orange line satisfies the constraint $y > x$, while the part "below" the orange line.

68. "Training" a SVM is the process of using the training data to learn the equation of the separator line shown in orange in this case. The feature vectors (shown as points in the figure) closest to the separator line on each side are called "support vectors." Once the separator line has been learned, classifying a new person

is easy as long as we have their associated feature vector. For instance, if a new person Tony has BMI = 25 and he exercises for 30 minutes per day on average, his feature vector would be (25,30) and his point in the 2-dimensional feature space would be depicted at (25,30). Therefore, Tony would be "above" the orange line as the *y* value (average exercise per day) is greater than his BMI, and he would be predicted to not be at risk of diabetes.

69. One of skill in the art will appreciate that as a SVM is trained or updated and new training data (support vectors) are added, the separator line changes to include the new training data. Further the support vectors change based on the changes to the separator line.

70. While the example below is for a SVM with a separator line that is straight, many types of separator lines in SVM exist, e.g., ones that do not require the line to be a straight line. Likewise, those skilled in the art will recognize that Figure 1 looks only at the case where there are two features. In cases where there are more than two features, the feature space is harder to visualize in print as is the separator line.

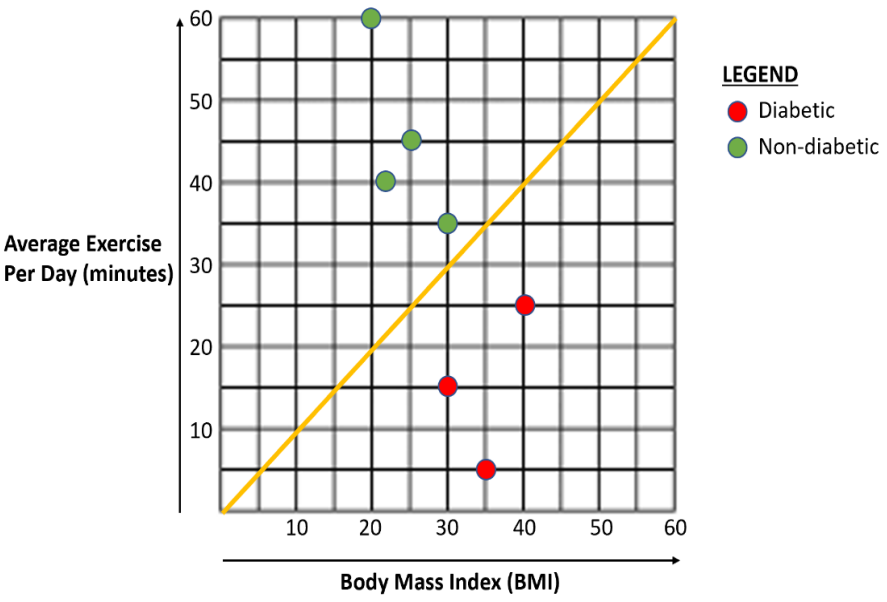
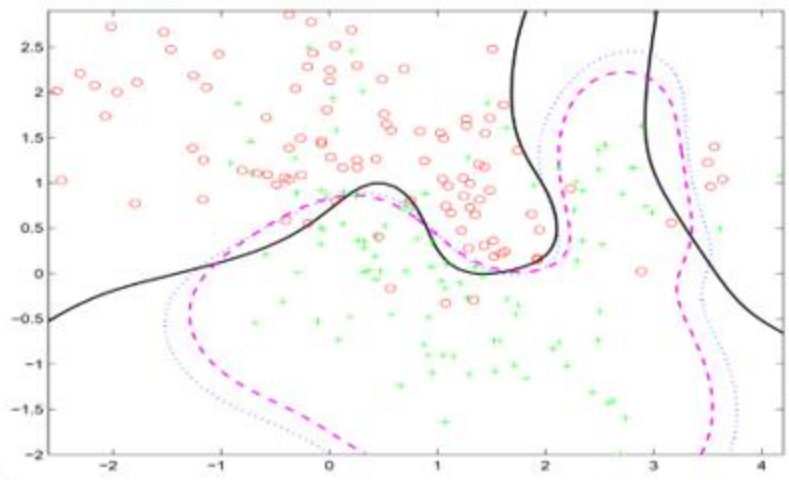


Figure 1

71. As indicated above, SVMs do not need to use a straight line for the separator line. As an example, Figure 2 below displays a pink dashed separator line based on an SVM with compactly supported kernel. Ex. 1012, p.12.



Classification boundaries given by the Bayes rule (solid line), SVM with Gaussian kernel (dotted line), and SVM with compactly supported kernel (dashed line).

Figure 2

VII. OVERVIEW OF THE PRIOR ART

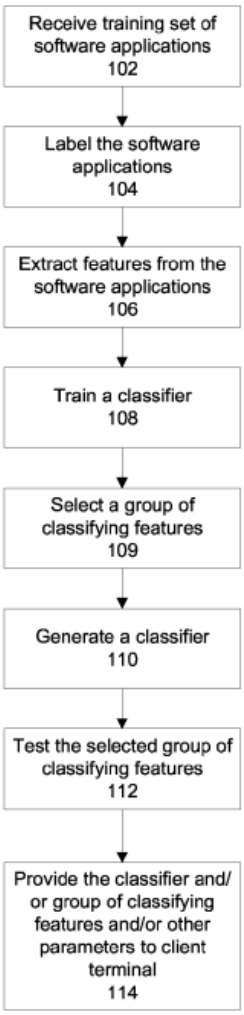
A. OVERVIEW U.S. PATENT APPLICATION PUBLICATION NO. 2015/0213376 TO IDESES ("IDESES")

72. Ideses is a U.S. Patent Application filed on January 30, 2014, and published on July 30, 2015.

73. Ideses describes "training a classifier" for classifying applications including, e.g., determining whether a software application is benign or whether it is harmful, malware, bothersome, adware, intrusive, spyware, or some other non-desirable classification type. Ex.1005, Abstract, ¶0023, ¶0141.

74. Ideses describes devices, methods, and systems to train a classifier and extract information from a software application (e.g., "**static meta data** regarding the application, for example, icon, name, rating or other features"). Ex. 1005, ¶0089 (emphasis added). Ideses further describes analyzing the extracted information with the classifier, and determining the application's classification type (e.g., "harmful") based on the analysis. *Id.*

75. Ideses describes, in Figure 1, an example of using data from multiple software applications as training data to train a classifier:



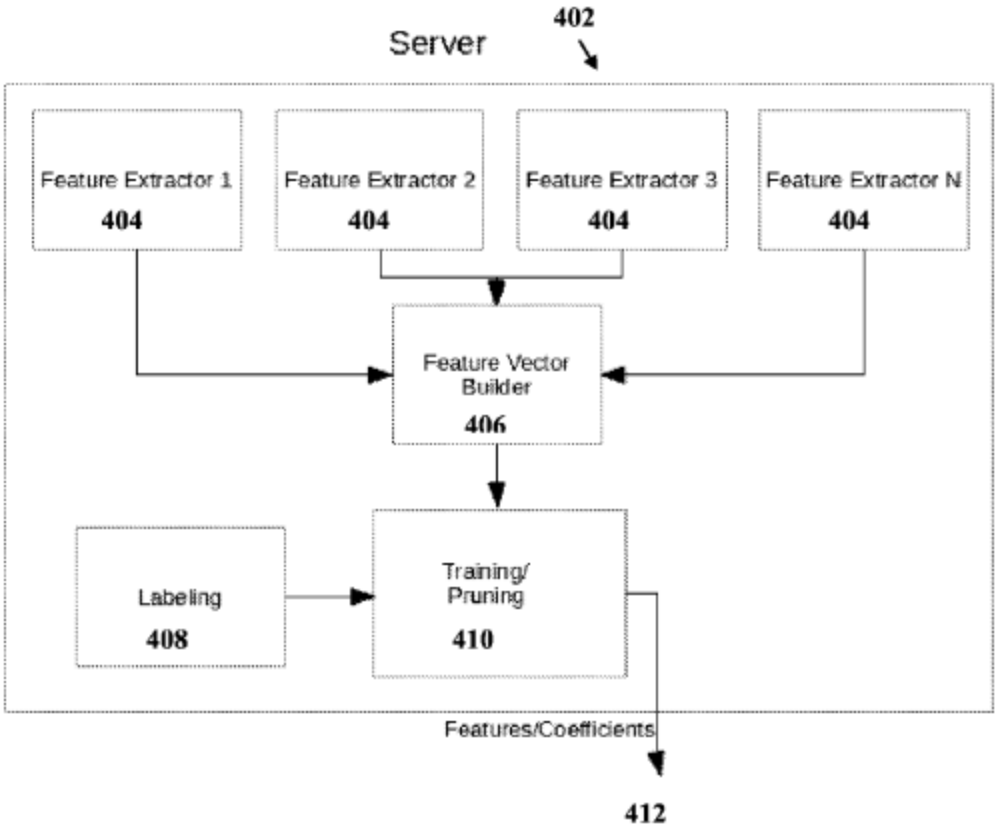
Ex. 1005, FIG. 1.

76. Ideses describes a central processor receiving data from multiple software applications for training the classifier, which could be provided manually by the operator (e.g., the manufacturer) or automatically downloaded. Ex. 1005, ¶0080.

77. Ideses further describes that the multiple software programs are labeled with a classification type based on desired software classification type categories (e.g., previously labeled applications that were vetted by commercial companies). Ex.

1005, ¶0081-¶0082. Ideses describes the desired software classification type categories as including labels such as desired or undesired software applications, adware, goodware, intrusive adware, or other types of classifications. *Id.* at ¶0081-¶0083.

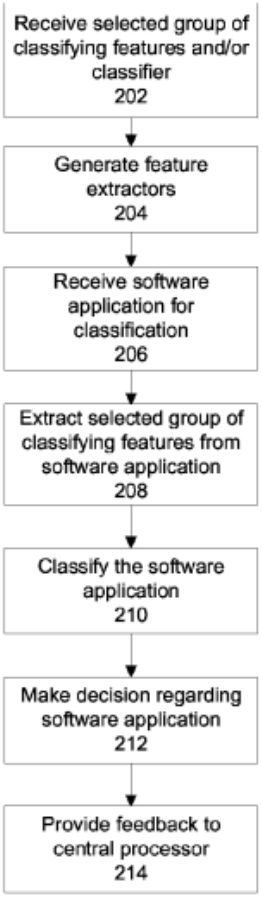
78. Ideses further describes, in Figure 1 above and in Figure 4 below, that features are extracted from the software applications, (e.g., static data points). Ex. 1005, ¶0086-¶0089. Ideses describes that any feature that describes the software application may be an extracted feature. *Id.* Ideses describes that the features extracted may be a partial set of features from software application or a complete set of features and said extracted feature values may be stored as a feature vector. *Id.*



Ex. 1005, FIG. 4.

79. Ideses describes that a classifier is trained on the sets of extracted features from the software applications and the labeling of the software applications. Ex. 1005, ¶0094. Once trained a trained classifier is generated. *Id.* at ¶0115.

80. Ideses describes, in Figure 2, using a generated trained classifier to classify a software application:



Ex. 1005, FIG. 2.

81. Ideses describes receiving (e.g., downloaded from the internet, loaded using physical computer readable media, uploaded by a third-party, etc.) and/or requesting installation a software application. Ex. 1005, ¶0131.

82. Ideses describes that features are extracted from the received software application (i.e., static data points). Ex. 1005, ¶0089, ¶0132. One of skill in the art would understand that the feature extraction for the newly received software application would be the same feature extraction performed on the training software applications.

83. Ideses further discloses that the extracted features are provided to the generated classifier and the classifier classifies the application (e.g., "as benign"). Ex. 1005, ¶0135. Once classified, a course of action is decided for the software application based on the classification (e.g., installation, "if the classification type is determined to be benign," or deleted, "if the classification type is determined to be harmful") *Id.* at ¶0141. Ideses further describes reporting the classification type outcome to a server. *Id.* at ¶0144

B. OVERVIEW OF U.S. PATENT APPLICATION PUBLICATION NO. 2016/0154960 TO SHARMA ("SHARMA")

84. Sharma is a U.S. Patent Application filed October 2, 2014, and published on June 2, 2016.

85. Sharma discusses systems, methods, and computer readable medium for training a risk rating framework (RRF) system for assessing the risk of a mobile application. Ex.1006, Abstract, ¶0024.

86. Sharma discusses using one or more classifiers to evaluate the risk and vulnerability of mobile computer applications. Ex.1006, ¶0024. The one or more classifiers use manifest and static features computationally extracted from the mobile computer application and indicates the probability of the application being malware based on those features. *Id.*

87. Sharma discusses, in Figure 1B, an overview of the RRF system:

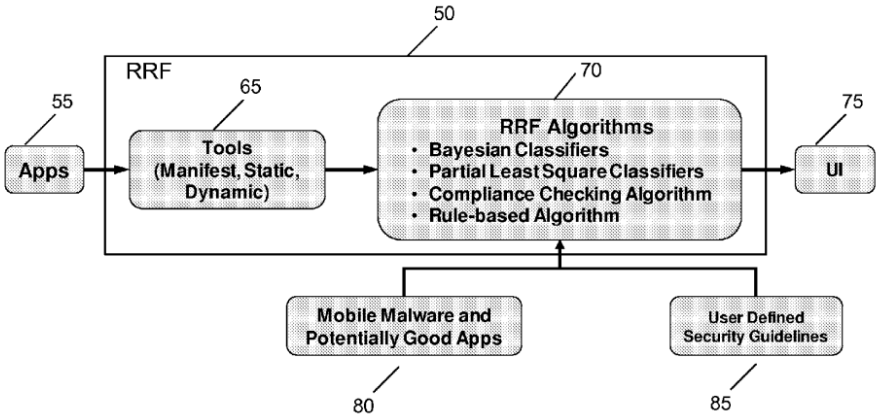


FIG. 1B

Ex. 1006, FIG. 1B.

88. Sharma discusses that classifiers (Ref. 70 in FIG. 1B) are used to analyze computer applications (Ref. 55 in FIG. 1B). Ex.1006, ¶0027. Sharma further discusses that tools (Ref. 65 in FIG. 1B) are programmed to extract certain features from computer applications (Ref. 55 in FIG. 1B) for analysis, which are used by classifiers to evaluate risk. *Id.*

89. Sharma discusses that some of the analysis tools include manifest, static, and dynamic analysis tools. Ex.1006, ¶0027. A manifest analysis tool may extract features and information from an application's manifest file, which contains information regarding files that are included in the computer application package. *Id.*

90. Sharma discusses that "[a] static analysis tool may extract features and information regarding an applications' software code, such as, code size, run time, and the like." Ex.1006, ¶0027. As I discussed above, static analysis refers to the process of examining code as it stands without running or executing the code.

91. Sharma discusses that "[a] dynamic analysis tool may extract features and information related to the computer application when it is running or operating, such as, incoming and outgoing application calls, and the like." Ex.1006, ¶0027.

92. Sharma discusses that analysis of manifest, static, and dynamic features can help identify computer applications that exhibit malware or malicious characteristics. Ex.1006, ¶0027

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

A. Ground 1: Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59 are obvious under 35 U.S.C. § 103 by Ideses

1. Claim 18

[18.0] A computer-implemented method comprising:

93. I understand that a preamble typically does not limit a claim. However, to the extent that the preamble is limiting, it is my opinion that it is disclosed by Ideses. One of ordinary skill in the art would understand "computer-implemented" to mean using a computer program on a computer.

94. Ideses is directed to "a method for training a classifier for classifying applications on a resource limited client terminal." Ex. 1005, ¶0006. From this statement alone, one of ordinary skill in the art would understand that the methods of training a classifier are implemented on a computer, as in the preamble of claim 1.

95. Ideses further describes that the "present invention may be a system, a method, and/or a computer program product" and that "the computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention." Ex. 1005, ¶0063. One of skill in the art would understand that since Ideses' method is implemented on a computer, such as a client terminal, Ideses discloses a computer-implemented method.

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

96. It is my opinion that Ideses discloses each part of this limitation.

97. Limitation [18.1] requires that static data points are "indicative of either a harmful or benign executable file."

98. Ideses describes that "software applications are labeled with a classification type based on desired software application classification categories." Ex. 1005, ¶0081. Ideses further describes "[f]or example, the classification types denote

desired or undesired software applications." *Id.* One of ordinary skill in the art would have understood the classification of desired or undesired software applications to be an indication of either a harmful or benign executable file.

99. At the time of the priority date of the '869 patent, the concept of executable files or software applications being labeled with a classification type was widely understood. For instance, in 2014, Markel and Bilzor stated that "[t]he features of each file are derived from some specific characteristics of the file, and each file is labeled as either benign or malicious." Ex. 1013, 1 (emphasis omitted). Likewise, Peiravian and Zhu in 2013 state, "[a]s a result of this this process, each App is represented as a single instance with binary permission and API call features, and a class label indicates whether the App is a benign or a malware." Ex. 1014, 303.

100. Limitation [18.1] further requires that "static data points" are identified. At the time of the priority date for the '869 patent, one of ordinary skill in the art would have understood the requirement of "identifying static data points" to mean that static characteristics of the executable file are extracted from the executable file without running or executing the file.

101. Ideses discusses that "features are extracted from the software applications." Ex. 1005, ¶0086. "[E]xtracted features may be any feature that describes the software application . . . . [T]he extracted features may contain static meta data

regarding the application, for example, icon, name, rating or other features." (static data points). *Id.* at ¶0089.

102. At the time of the priority date of the '869 patent, a person of ordinary skill in the art would have understood the expression "features are extracted from the software applications" to mean that certain static characteristics of the applications were extracted from the application files without the need to run or execute the application.

103. In one example, Markel and Bilzor illustrated that this concept was readily understood in 2014 in stating, "[w]e used a Python module called pefile [1] to extract 44 different features from the PE32 headers, plus 2 boolean features derived from the section entropies of a PE32 file, to build a database. Each file in the database had a record containing the features and the label, benign or malicious, of that file." Ex. 1013, 2.

104. The term "static" meta data would have referred to features or characteristics of a software application that could be extracted without running the program. This is illustrated by Gandotra et al. in 2014 who state that "[a]nalyzing malicious software without executing it is called static analysis." Ex. 1015, 57.

105. Further, Peiravian and Zhu further illustrate this concept in 2013 by stating, "[o]ne of the most popular behavioral methods is malware detection based on static requested permissions, which check what types of resources, such as Wi-Fi network,

user location, and user contact information, an App is requesting for installation (Android provides over 130 permissions for developers to control the resources that an App can request [16]).” Ex. 1014, 301.

106. As I indicated above, *static* analysis of an executable file would be information about the file that can be obtained without executing or running the file. Accordingly, both static data points and static meta data would be information that can be obtained from the executable file without running/executing the file.

107. Accordingly, one of ordinary skill in the art would have understood extracting features to be the same as *identifying* the values of features. Further one of skill in the art would have understood *static* meta data to be *static data points*.

108. Limitation [18.1] further requires that the identified “static data points” “may be indicative of either a harmful or benign executable file.” At the time of the priority date for the ‘869 patent, one of ordinary skill in the art would have understood the above requirement to mean that static characteristics of the executable file are used to categorize the executable file as harmful or benign.

109. Ideses discusses that “[m]odules 314 may include a run-time feature extractor for extracting features from a Software application for classification based on the selected group of classifying features.” Ex. 1005, ¶0076. For example, “a complete set of features is extracted from each application. Alternatively, individual or groups of features are extracted from each application, for example, as features are being

evaluated for inclusion in the group of classifying features" (indicative of either a harmful or benign executable file). *Id.* at ¶0086.

110. At the time of the priority date of the '869 patent, a person of ordinary skill in the art would have understood Ideses to discuss assessment of a set of features to determine how effective the set of features were in helping to distinguish harmful executable files from benign ones.

111. This concept was well known at the time, as illustrated by Islam et al. in 2013 who reference a 2001 paper when stating "[c]onventional malware detection and classification systems are based on static features extracted from executables by reverse-engineering. An earlier paper (Schultz et al., 2001) introduced the static feature extraction process based on three different types of features: Portable Executable (PE), byte-sequence *n*-grams, and string features. In the PE approach, the features are extracted from dynamic link library information inside PE executables stored in Win32 PE binaries. The byte sequences (*n*-grams) approach uses sequences of n bytes extracted from an executable file; this type of feature was shown to yield high accuracy in detecting new malware in Schultz et al. (2001). In Abou-Assaleh et al. (2004), Kolter and Maloof (2004), and Ye et al. (2010), the authors used the *n*-gram technique to deal with two classification problems: (1) classification between clean and malicious executables, and (2) classification of executables as a function of their payload." Ex. 1016, 647. This clearly shows a

number of groups of features (e.g., PE features, byte sequence features and *n*-gram features. *Id.*

112. Accordingly, one of ordinary skill in the art would have understood Ideses to disclose identifying a set of static features (i.e. points of data obtained without the need to run or execute the file) associated with an executable file and using those static features to train a classifier to determine if an executable file is harmful or benign.

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

113. It is my opinion that Ideses discloses each part of this limitation.

114. Limitation [18.2] requires a plurality of categories of files including harmful files and benign files. Ideses discloses that, "multiple software applications are labeled with a classification type based on desired software application classification categories. For example, the classification types denote desired or undesired software applications" (plurality of categories of files including harmful files and benign files). Ex. 1005, ¶0081. In another example, Ideses teaches "the classification type is benign or adware." *Id.* at ¶0023; see also *id.* at ¶0083, ¶0135, ¶0141.

115. Such classification types were well known in the art at the priority date of the '869 patent. For example, as early as 2007, Cai et al. discussed "the use of byte

sequence frequencies as a way to automatically distinguish malicious from benign executables without actually executing them." Ex. 1017, 3156. Adding that, "[w]e found that single-byte patterns provided surprisingly reliable features to separate malicious executables from benign. Between classifiers and feature selection methods, the overall performance of the models depended more on the choice of classifier than the method of feature selection. Support vector machine (SVM) classifiers were found to be superior in terms of prediction accuracy, training time, and aversion to overfitting." *Id.*

116. Limitation [18.2] further requires associating the identified static data points with one of the categories. Ideses discloses that the extracted features (as I explained earlier, equivalent to static data points) are associated with one of the classification types (as I explained earlier, equivalent to categories of files) using a classifier. For example, "a complete set of features is extracted from each application. Alternatively, individual or groups of features are extracted from each application, for example, as features are [sic] being evaluated for inclusion in the group of classifying features." Ex. 1005, ¶0086.

117. At the time of the priority date of the '869 patent, the idea of evaluating features for inclusion in the group of classifying features was well-known and well understood. It was well-known and well understood that performance (in terms of predictive accuracy in distinguishing malware from benign apps) could increase if a

subset of a large set of features was used. For instance, Yan et al. in 2013 state "[w]e conduct a systematic study on the discriminative power of various types of features extracted from malware programs, and experiment with different combinations of feature selection algorithms and classifiers." Ex. 1018, 41.

118. Ideses further discloses that "the features are stored as feature vectors, which may be used for training the classifier." Ex. 1005, ¶0091. One of skill in the art would understand that the disclosed "training the classifier" in Ideses is equivalent to the claimed associating. At the time of the priority date of the '869 patent, the list of features used to distinguish malicious apps from benign ones were typically stored as a vector form (similar to an ordered list).

119. Ideses also explains an association of extracted features (as I explained earlier, equivalent to static data points) using classification labeling (as I explained earlier, equivalent to harmful files and benign files). For instance, "a classifier is trained based on the set of extracted features (block 106) and software classification labeling (block 104)." Ex. 1005, ¶0094. "A single classifier or multiple classifiers may be used. For example, a combination of classifiers may be applied to classify feature vectors, for example a cascade of classifiers, a boosting topology of classifiers, or a parallel classification scheme." *Id.*

120. Moreover, one of ordinary skill in the art would understand that, as discussed for example, in Islam et al. in 2010, "[o]ur framework combines the static features

of function length and printable string information extracted from malware samples into a single test which gives classification results better than those achieved by using either feature individually. In our testing we input feature information from close to 1400 unpacked malware samples to a number of different classification algorithms." Ex. 1019, 9; *see also* Ex. 1020 (discussing using use multiple different classifiers that will classify a new instance predicting its real class). Thus, using feature information for classification was well-known.

[18.3] identifying an executable file to be evaluated;

121. It is my opinion that Ideses discloses each part of this limitation.

122. Limitation [18.3] requires identifying an executable file. Ideses discloses identifying a "software application." Ex. 1005, ¶0131. As I have explained, low-level programs are often called executables, which is used interchangeably with "software application."

123. Ideses further discloses that the software application is "received at the mobile device, for example, the software application is downloaded from the internet, loaded using physical computer readable media, uploaded by a third party, or other methods of receipt." Ex. 1005, ¶0131. Such downloading, using physical computer readable media, uploading, or other methods of receipt are examples of identifying the software application.

124. At the time of the ‘869 patent, a software application was similar to a mobile phone or laptop/desktop “app” as an executable piece of code that can be installed in the machine in one of many ways. As one example, it could be downloaded (received) onto the device either explicitly by the user, or by an operating system or program with or without explicit user involvement. The app could be loaded using physical media (e.g., by plugging a phone into a laptop), or placed on the device by a third party (e.g., a malware developer).

125. Limitation [18.3] further requires that the executable file is to be evaluated. Ideses discloses, for example, the software application is classified “by a run-time classification module 314.” Ex. 1005, ¶0135. In a further example, “the software application is classified, for example, as benign or adware.” *Id.* The disclosed classification in Ideses is an evaluation.

126. As I have explained earlier, a machine learning algorithm (also called a classifier) is used to learn a “model” or “predictive model” from the training set. Such a model (once learned or trained) can automatically identify a condition on the features such that when an object’s feature vector satisfies the condition, it is very likely that the object belongs to one class (e.g. person it at risk of diabetes, or application is malware) and when the object’s features do not satisfy the condition, the object very likely belongs to the other class (e.g. person is not at risk of diabetes, or application is benign).

127. Just as classifiers can classify apps (executable files) into benign or malicious classes, they can do the same in terms of classifying apps as benign or adware. As stated by Ideses and Neuberger in 2013, "[a]dware are essentially applications whose main purpose is getting maximal revenue to the developer while giving the user the minimal amount of value." Ex. 1021, 1.

128. As an example of separating benign apps from adware, as is known by one of ordinary skill in the art, Shahzad et al. (2011) state "[t]he main contributions of the paper is a large publicly available adware data set, an accurate adware detection algorithm, and an extensive empirical evaluation of several candidate machine learning techniques that can be used in conjunction with the algorithm." Ex. 1022, 189.

[18.4] extracting a plurality of static data points from the executable file;

129. It is my opinion that Ideses discloses each part of this limitation.

130. Extraction of features from an executable (malicious or benign) is an essential part of a machine learning process. Features are characteristics of a software application and are often extracted directly from the application itself via a number of methods such as static and/or dynamic analysis of the application as I have explained earlier in my discussions of machine learning and static and dynamic malware analysis.

131. Ideses discloses that "features are extracted from the software applications," and that "extracted features may be any feature that describes the software application." Ex. 1005, ¶0086, ¶0089. As one example, extracted features may include the size of the app, whether the app is signed or not, the types of permissions that the app requests, and so on.

132. Ideses further discloses that "features are extracted based on the software application, for example by the generated feature extraction modules (block 204)." Ex. 1005, ¶0132. "[T]he features extracted are based on the group of classifying features and/or trained classifier received from the central server (block 202)." *Id.* "[T]he features are extracted by the generated feature extractors (block 204)." *Id.* Ideses further discloses, "[o]ptionally, a learning module 306 (e.g., stored on memory 304) of a machine learning algorithm is applied to train the classifier." *Id.* at ¶0094. Ideses thus discloses using a classifier (machine learning algorithm) for generating the feature vector because generating a feature vector is a necessary step in training a classifier, as I have previously explained in my discussion on machine learning models.

133. One of ordinary skill in the art would understand that extractors are pieces of software that automatically extract the values of features of a software application. An example of such a feature extractor is discussed in Canzanese et al. (2013) who state: "[t]he feature extractor processes the raw data and sends the extracted features

to a malware detector. If malware are detected, the malware classifier is used to classify the malware, and the identifying information output by the classifier is used to determine appropriate mitigation." Ex. 1023, 113.

134. In yet another example, one of ordinary skill in the art would understand, Ye et al. (2010) describe a system called AMCS and state "AMCS first uses the feature extractor to extract the function-based instructions from the collected PE malware samples, converts the instructions to a group of 32- bit global IDs as the features of the data collection, and stores these features in the signature database." Ex. 2024, 97.

135. With reference to the plurality of static data points, the '869 patent describes "[a] static data point may be a point of reference used to evaluate an executable file. As examples, static data points include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples." Ex. 1001, 5:61-67. Other examples are illustrated in TABLE 1.1:

TABLE 1.1

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

Ex. 1001, TABLE 1.1

136. Ideses discloses such static data points. For example, "the extracted features may contain static meta data regarding the application." Ex. 1005, ¶0089. Static meta data regarding the application is an example of static data points from the executable file.

137. In specific examples, Ideses discloses, "the extracted features may contain information regarding the executable code and/or software package, for example, in the form of byte code, resources permissions or other features." Ex. 1005, ¶0089; *see also* Ex. 1001, 5:61-67, TABLE 1.1.

138. Ideses' disclosed "byte code" is equivalent to the example "package code" or "product code" in TABLE 1.1 of the '869 patent. Ideses' disclosed "resources" is equivalent to the example "resources" or "resource information" in TABLE 1.1 of

the ‘869 patent. Ideses’ disclosed resources permissions is equivalent to the example “certificate information” in TABLE 1.1 of the ‘869 patent.

139. As I have explained earlier in my discussion of static and dynamic malware analysis, association of a set of static features with any software application or executable code was well known at the time of the priority date of the ‘869 patent. *See* Ex. 1025.

140. In another example, the permissions sought by a software application are also frequently used in order to detect malware because both benign and malicious software applications may need certain permissions in order to perform their functions. Ex. 1026, 81 (“apply static analysis with the aim to understand the significance of permissions for the identification of unseen samples. Investigation is carried out to identify the most prominent permissions that contribute in mobile malware classification with few features.”).

141. Thus, Ideses discloses extracting a plurality of static data points from an executable file, which was well-known in the art, exemplified by the other examples presented.

[18.5.a] generating a feature vector from the plurality of static data points

142. It is my opinion that Ideses discloses each part of this limitation.

143. Ideses discloses "generating a selected group of classifying features (e.g., feature vector) for classification of software applications." Ex. 1005, ¶0071. As I have previously explained, static analysis typically refers to the process of examining code as it stands, while dynamic features are obtained by running the application, typically within a controlled environment that is often called a "sandbox." Generating a feature vector corresponds to the process of identifying a set of features that may be associated with a software application and extracting the values of those features for a given software application. As discussed in Ideses, such features may include static features or dynamic features or a mix of both.

144. Ideses also discloses that the extracted features are collated into feature vectors. Ex. 1005, ¶0134. As I have previously explained, the extracted features can be static data points. Moreover, as I have further explained, static data points are the values of static features. A feature vector is an ordered list of values associated with a software application where the first value corresponds to the value of the first feature, the second value is the value of the second feature, and so forth.

[18.5.b] using a classifier trained to classify the static data points based on training data,

145. It is my opinion that Ideses discloses each part of this limitation.

146. Ideses discloses that a classifier is trained. For example, "at 108, a classifier is trained based on the set of extracted features (block 106) and software

classification labeling (block 104). Optionally, a learning module 306 (e.g., stored on memory 304) of a machine learning algorithm is applied to train the classifier." Ex. 1005, ¶0094. Ideses continues to explain that "at 202, the group of classifying features, the trained classifier, and/or the pruned list of features (e.g., feature vector) is received by the mobile device." *Id.* at ¶0124.

147. As I have explained, and is exemplified by Ideses as cited above, a machine learning algorithm trains a classifier (or analogously, learns a model). Such a model learns a condition on the features such that when the feature vector of an object (e.g. a software application) satisfies the condition, there is a high likelihood of the object belonging to one class (e.g. malware), whereas when the feature vector of the object does not satisfy the condition, there is a high probability of belonging to another class (e.g. benign application).

148. Ideses discloses that the features are extracted based on a trained classifier. For example, as I have explained with respect to section [18.5.a], Ideses discloses generating a feature vector. Further, Ideses discloses, "the features extracted are based on the group of classifying features and/or trained classifier received from the central server (block 202)." Ex. 1005, ¶0132. Thus, one of ordinary skill in the art would have understood that the feature vector is generated depending on the results of training the classifier based on training data because the features extracted are a result of the previous training.

149. In another example, Ideses discloses "[a] classifier is trained by a training module 410." Ex. 1005, ¶0153. As I have previously explained, "training" a classifier is the process of using the training data to learn, for example, the equation of a separator line, as in my earlier example. Therefore, the features extracted based on the trained classifier necessarily use training data to train the classifier.

150. Ideses further discloses that features extracted to generate the feature vector are based on the trained classifier to classify the static data points. For example, Ideses discloses "[t]he group of classifying features has been selected based on the classification effectiveness and/or based on resource requirements of each feature." Ex. 1005, ¶0124; *see also id.* at ¶¶0071, 0132. Accordingly, the feature vector is tailored based on feature selection in view of how accurately certain features make classifications (in other words, classify the static data points).

151. One of ordinary skill in the art would have recognized that the feature vector is generated based on the described extraction and/or selection in Ideses. As I have explained, the feature vector is therefore generated based on each of the features, and using a classifier trained to classify the static data points.

152. One of ordinary skill in the art would have readily expected success in the disclosed trained classifier to classify the static data points based on training data as exemplified in Ideses because of the known steps of machine learning, as I have explained above in paragraphs 59-71. More particularly, the process of defining a

set of features and operating on the set of features to generate classifying training data is the use of known components in accordance with their conventional uses in a predictable art.

153. Ideses further discloses "[o]ptionally, a learning module 306 (e.g., stored on memory 304) of a machine learning algorithm is applied to train the classifier" and references supervised and unsupervised learning modules for training the classifier. Ex. 1005, ¶0094; *see also id.* at ¶0096.

154. One of ordinary skill in the art would have recognized that training a classifier includes generating feature vectors to train the classifier. Machine learning classifiers take a training set as input. The training set consists of a set of training samples (apps in the case of the Ideses and '869 patent). Each training sample consists of a feature vector and a class label (e.g., whether the app is benign or malicious). The feature vector is typically generated via software code that extracts the values of each feature from the app. For instance, a piece of code might generate the value of the feature "size" of the app (e.g., the value 900 Kilobytes). Machine learning algorithms cannot train a classifier without such feature vectors.

155. Ideses discloses "a machine learning algorithm is applied to train the classifier." Ex. 1005, ¶0094. Accordingly, one of ordinary skill in the art would have recognized that the machine learning algorithm needs training data using which it can train the classifier.

156. That training data is comprised of a set of training samples. Each training sample consists of a feature vector and a class label. Without such feature vectors, there is nothing to learn from as one would be left merely with a set of apps designated as benign or malicious with no additional information about the apps to use as a basis for learning a classifier that can distinguish between malicious and benign apps.

157. One of ordinary skill in the art would have further recognized that a classifier is a machine learning algorithm. While there are many forms of machine learning, classifiers are the most widely used form of machine learning. Classifiers learn conditions on feature values that enable them to distinguish feature vectors associated with one class from another class.

158. Accordingly, one of ordinary skill in the art would have found using a classifier trained to classify the static data points based on training data obvious based on the disclosure in Ideses.

[18.5.c] the training data comprising files known to fit into one of the plurality of categories of files,

159. It is my opinion that Ideses discloses each part of this limitation.

160. Ideses discusses, "at 102, **multiple software applications for *training* the classifier** are received, for example, by central processor." Ex. 1005, ¶0080 (emphasis added). "The software applications may be manually provided by an

operator of the system (e.g., the manufacturer), provided by software application manufacturers, automatically downloaded from the internet, provided by updates from mobile devices." *Id.* The described multiple software applications reflect the training data.

161. In particular, Ideses continues that "at 104 the multiple software applications are labeled with a classification type based on desired software application classification categories. For example, the classification types denote desired or undesired software applications" (fit into one of the plurality of categories of files). Ex. 1005, ¶0081. "Examples of labels include: Adware, Goodware, Intrusive Adware, or other classification types may be used." *Id.* at ¶0083.

162. At the time of the '869 patent, software applications could be labeled with many different classification types. A simple classification might be "benign" or "malicious" as I have described above. Other types might classify malware into specific families.

163. For instance, Rieck et al. in 2007 described techniques to identify one of 14 malware families a given software application belongs to:

1: Backdoor.VanBot	(169)	8: Worm.Korgo	(4)
2: Trojan.Bancos	(208)	9: Worm.Parite	(19)
3: Trojan.Banker	(185)	10: Worm.PoeBot	(188)
4: Worm.Allaple	(614)	11: Worm.RBot	(904)
5: Worm.Doomber	(0)	12: Worm.Sality	(0)
6: Worm.Gobot	(0)	13: Worm.SdBot	(597)
7: Worm.IRCBot	(107)	14: Worm.Virut	(144)

Ex. 1027, Table 2.

[18.5.d] wherein one or more features of the feature vector are selectively turned on or off

164. It is my opinion that Ideses discloses each part of this limitation.

165. For example, Ideses discloses "selecting a set of classifying features extractable from software applications, in order to generate classifiers, provide the selected group of classifying features (e.g., feature vectors), select coefficients and/or other parameters for classification of software applications. Optionally, the selected group of classification features is a sub-set of the available features (all or some) that may be extracted from a software application. The complete set of features may be pruned to generate the selected group." Ex. 1005, ¶0044. Ideses also explains that "features (or their equivalents) that do not contribute (or do not significantly contribute) to the classification process are pruned or not included." *Id.* at ¶0107. Accordingly, features that are pruned or not included are selectively turned off in the feature vector.

166. As I have explained earlier, much research has gone into identifying the best subset of features from a larger set of features. The "best subset" would be the subset of a given set of features that generate the highest prediction performance. Once the best features are identified, the remaining (not best) features are eliminated, and the surviving features are used for prediction. *See* Ex. 1028, 177 ("[t]he aim of our work

is to show empirically that a significant reduction in the number of features is possible while at the same time maintaining a good level of accuracy.”).

167. Ideses further discloses “[a]lternatively or additionally, each feature of the group is selected from-the-ground-up; added to the group as opposed to removed from or retained within a larger group.” Ex. 1005, ¶0071. Adding new features to a given suite of features has been a common aspect of machine learning research before the priority date of the ‘869 patent. For instance, my own work illustrated by the below example on analyzing Twitter bots exemplifies adding features to a given feature group. Ex. 1029

In this section, we define the profile of a user that is used by SentiBot. The measures are divided into four categories, each described in a subsection below.

1) *Tweet Syntax:* This class of variables captures various aspects of the construction of tweets. *This class of variables has been used before (see, e.g., [5]) and is not claimed as a novel contribution.*
2) *Tweet Semantics:* This class of variables captures various aspects of the content of tweets from a semantic and sentiment orientation and how they have changed over time. *This class of contextual variables, as well as (3) and (4) below, consists of several sentiment-related variables that have never been used before.*
3) *User Behavior:* This class of variables captures various aspects of the user (derived from his Twitter profile when available) as well as several sentiment-related variables showing the user's sentiments and their variance over time.
4) *User Neighborhood:* This class of variables captures various aspects of the immediate neighborhood around the user, together with what those neighbors are discussing and what sentiments they hold on various topics. It includes several sentiment-related contextual variables that capture measures of the user's sentiments (on various topics) as compared to those of his neighbors.

Ex. 1029.

168. Accordingly, features that are selected from the ground up or added to the group are selectively turned on in the feature vector.

[18.5.e] based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

169. It is my opinion that Ideses discloses each part of this limitation.

170. Ideses discloses "features are selected based on an identified classification effectiveness rank for each of the features." Ex. 1005, ¶0045. One skilled in the art

would understand that the described selection is an evaluation of a value of one of the plurality of static data points.

171. Ideses further discloses "classification effectiveness sometimes refers to the contribution of the extracted feature to the accuracy of classification." Ex. 1005, ¶0045. "[I]f the accuracy of classification with the feature is 95%, and the accuracy of classification without the feature is 50%, the accuracy in classification based on the feature is 45%, which may [be] significant for classification effectiveness." *Id.* "Significant features, such as those that have a large effect on the classification outcomes, may be retained. Insignificant features, such as those that have a small, negligible or no effect on classification outcome may be pruned." Ex. *Id.* at ¶0106. One skilled in the art would understand that the significance for classification effectiveness, as exemplified by the example ranges is whether a value of one of the plurality of static data points is within a predetermined range.

172. As of the priority date of the '869 patent, the well-known technique for assessing the importance of a feature was the one described by Ideses above. Given a set F of features, the importance of a specific feature f for a predictive task was measured by assessing the predictive accuracy with the feature (e.g., 95%) and without (e.g. 50%) and the different (45% in our example) was viewed as the contribution of the feature to the learned predictive/classification model.

173. In another example of an evaluation of a value of one of the plurality of static data points, Ideses discloses, "the group of classifying features is selected to reduce the dimensionality of the feature vector." Ex. 1005, ¶0105. Continuing the example, as an example of whether a value of one of the plurality of static data points is within a predetermined range, Ideses discloses "features that correspond to coefficients having a value of zero are pruned or not included" *Id.* at ¶0107. Selection of the group of classifying features based on coefficients with zero value may be denoted by the expression: $\text{Feature}_i \epsilon FVif|(\text{Coefficient}_i)| \neq 0$. *Id.* at ¶0109. One skilled in the art would understand that coefficients having a value of zero are accordingly within a predetermined range, as required by the claim.

174. In another example of whether a value of one of the plurality of static data points is within a predetermined range, Ideses discloses that "**features** with a **low magnitude value of the coefficient are identified for pruning** and/or are not included." Ex. 1005, ¶0107 (emphasis added). Selection of the group of classifying features based on coefficients with low values may be denoted by the expression: $\text{Feature}_i \epsilon FVif|(\text{Coefficient}_i)| \geq \alpha$. *Id.* at ¶0110.One skilled in the art would understand that pruning features according to a low magnitude value is evaluation of a value within a predetermined range.

175. Ideses discloses at least three examples of features (e.g., *static data points*) evaluated for inclusion (e.g., *selectively turned on or off*) based on whether a value

of one of the features is within a predetermined range. In the case of specific classifiers such as a support vector machine, as I have explained above, the separator line may have an equation such as $3x_1 + 4x_2 + 5x_3 = 4$. Here x_1, x_2, x_3 are the values of 3 features. One of ordinary skill in the art would understand that such separator lines can be easily defined when there are more than 3 features being extracted. When an app's feature vector satisfies the condition $3x_1 + 4x_2 + 5x_3 > 4$, the app might be declared to be malware (for example purposes only). When the app's feature vector does not satisfy this condition, it might be declared to be benign. In this case, the coefficients of the 3 features are 3, 4, 5, respectively. Features with very small coefficients were often considered very unimportant. For instance, if the first term in the separator line's equation had been $0.00001x_1$, for example, if the coefficient of x_1 had been 0.00001 instead of 3, then a feature selection algorithm might simply have considered the first feature to be unimportant.

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

176. It is my opinion that Ideses discloses each part of this limitation.

177. In general, once a classifier has been trained to distinguish between two classes (e.g., malicious vs. benign software applications), any new software application can be classified by applying the trained classifier to the feature vector of the new (i.e., not previously seen during training) application. "Evaluating" refers

to the process of checking whether the feature vector of the new software application satisfies or does not satisfy the previously learned condition. Depending on the outcome of this "check," the classifier will label the software application as either "malicious" or "benign."

178. As of the priority date of the '869 patent, it was well understood that a feature vector was just an ordered list of feature values. Such feature vectors may be stored in many different types of storage including buffers, random access memory or RAM, hard disk, and other storage devices. For example, Ideses discloses that the "extracted features are collated into feature vectors. The feature vectors may be stored in buffers, which may provide for easy serial access." Ex. 1005, ¶0134. I also refer to my analysis in [18.5.a] - [18.5.e].

179. Ideses also discloses evaluation of the feature vectors using a machine learning module. "[T]he software application is classified based on the group of classifying features that have been computed by the feature extractor (block 208)." Ex. 1005, ¶0137. Ideses discloses that "features that correspond to coefficients having a value of zero are pruned or not included, for example, when classification is performed based on a support vector machine (SVM)." *Id.* at ¶0107.

180. Ideses discloses "[a]t 210, the software application is classified, for example, by a run-time classification module 314 stored on memory 312." Ex. 1005, ¶0135. "[T]he classification module labels the software application." *Id.* "[T]he software

application is classified, for example, as benign or adware." *Id.* I refer also to my analysis in section [18.5.c].

181. As of the priority date of the '869 patent, it was common to first learn a classifier (i.e. a predictive model) by going through the training process, and then use the learned model, in an operational phase, to make predictions when new data (e.g. a new software application) is presented to the learned model. Accordingly, Ideses discloses that an evaluation is performed using the SVM for classification of a software application as malicious or benign.

182. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 18 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 18 is rendered obvious over Ideses.

2. Claim 20

[20.0] The computer-implemented method according to claim 18,

183. See [18.0] - [18.6].

[20.1] wherein the machine learning model comprises an artificial neural network.

184. It is my opinion that Ideses discloses the additional limitations of dependent claim 20.

185. Ideses discloses the use of machine learning models. For example, "the learning module performs the machine learning and/or classifier training." Ex. 1005, ¶0095. I refer also to my analysis in section [18.6]. Ideses discloses that specific "[e]xamples of software modules to train the classifier include: *Neural Networks.*" Ex. 1005, ¶0096 (emphasis added).

186. A neural network (or artificial neural network) is a complex structure that can be used for classification, as well as for a variety of other purposes. Neural networks have been used for malware detection since before the priority date of the '869 patent. *See* Ex. 1030, 100 (discussing an "Artificial Neural Network (ANN)-based malware detection system that uses both permissions and system calls to detect unknown malware. In our system, we consider two types of ANNs: Feedforward Neural Networks (FNN) to learn the patterns of permissions and Recurrent Neural Networks (RNN).").

187. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 20 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known

components in accordance with their conventional uses in a predictable art. Thus, claim 20 is rendered obvious over Ideses.

3. Claim 22

[22.0] The computer-implemented method according to claim 18,

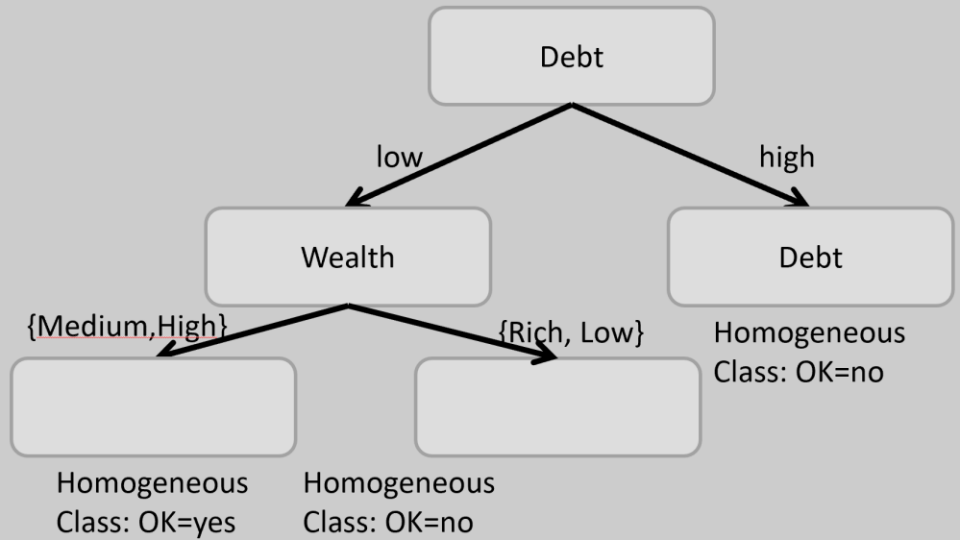
188. See [18.0] - [18.6].

[22.1] wherein the machine learning model comprises a machine learning decision tree.

189. It is my opinion that Ideses discloses the additional limitations of dependent claim 22.

190. Ideses discloses the use of machine learning models. For example, "the learning module performs the machine learning and/or classifier training." Ex. 1005, ¶0095. I refer also to my analysis in section [18.6]. Ideses discloses that specific "[e]xamples of software modules to train the classifier include: . . . *Decision Trees*." Ex. 1005, ¶0096 (emphasis added). One skilled in the art will readily appreciate that a decision tree is the claimed machine learning decision tree.

191. A decision tree is a classifier that uses a tree representation such as the one shown below.



192. Each interior node (i.e., node with two or more children) in the tree evaluates a condition on the features and creates two or more children corresponding to different ways of satisfying the condition. In this example, the root node which is an interior node looks at a feature called "Debt" and has two children, one corresponding to feature vectors with "low" debt and another corresponding to feature vectors with "high" debt. The interior node labeled "Wealth" looks at a feature called "Wealth" and branches left if the value of that feature is in the set {Medium, High} and branches right if the value of that feature is "Low." While I show this example for illustrative purposes only with a very small set of features, one of ordinary skill in the art at the priority date of the '869 patent would recognize that a more sophisticated and fleshed out version of such a decision tree might be used, for instance, to consider applicants for a loan or a credit card. In such a decision

tree, the path from the root of the tree to each node in the tree corresponds to a logical condition.

193. For instance, the path from the root node (labeled “Debt” to the “Wealth” node to the node with no label at the bottom center of the figure corresponds to the logical condition (“Debt = Low & Wealth is in the set {Rich, Low}). If the feature vector associated with a loan or credit card applicant satisfies this logical condition, then that person is denied a loan according to this particular decision tree. This example is purely for illustrative purposes and not for actual approval or denial of a loan or credit application. Each leaf node (i.e., node with no children) such as the node at the bottom left of the figure shown above is considered to be sufficiently homogeneous in the sense that there is no need to further split that node into children.

194. Training a decision tree model involves understanding how to learn a decision tree from a body of training data. Operationally, using a previously learned decision tree involves taking the feature vector of a new object to be classified and finding the unique path from the root of the tree to a leaf node corresponding to that feature vector. The label of the dominant class at that node is then assigned to the new feature vector as the predicted label. If we look at this decision tree and consider a hypothetical loan applicant Mary who has low debt and whose wealth is “high”, this decision tree would approve her for a loan by following the leftmost path in the decision tree shown that leads to a homogenous node (i.e. all training data whose

feature vectors satisfied the condition along the leftmost path belonged to the class "yes", i.e. approve).

195. Decision trees were used extensively for classification of software applications into malicious vs. benign categories prior to the priority date of the '869 patent. See Ex. 1031, 42 ("Among the four classifiers we have tested in this study, we find that a variant of the decision tree classifier (i.e., C4.5) performs consistently well in classifying all malware families. The decision tree classifier is known to have scalability advantages over other classifiers such as SVM and kNN.").

196. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 22 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 22 is rendered obvious over Ideses.

4. Claim 24

[24.0] The computer-implemented method according to claim 18,

197. See [18.0] - [18.6].

[24.1] wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

198. It is my opinion that Ideses discloses the additional limitations of dependent claim 24.

199. Ideses discloses "extracted features are collated into feature vectors. The feature vectors may be stored in buffers, which may provide for easy serial access" and "the software application is classified based on the group of classifying features that have been computed by the feature extractor (block 208)." Ex. 1005, ¶ 0137, *see also Id.* at ¶0134. The software application being classified based on the group of classifying features that have been computed by the feature extractor is evaluating the feature vector. I refer also to my analysis in section [18.6].

200. Ideses also discloses, "features that correspond to coefficients having a value of zero are pruned or not included, for example, when classification is performed based on a support vector machine (SVM)." Ex. 1005, ¶0107. As I explained earlier, in the case of specific classifiers such as a support vector machine (referring also to my analysis in section [18.5.e]), the separator line may have an equation such as $x_1 + x_3 + 2x_5 = 30$ involving 5 variables x_1, x_2, x_3, x_4, x_5. However, x_2, x_4 have zero coefficients and hence are not explicitly shown in this equation. This limitation suggests that these two features be dropped when using the classifier because their coefficients are zero. Accordingly, classification performed based on

a support vector machine (SVM) as disclosed in Ideses is evaluating the feature vector using support vector processing.

201. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 24 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 24 is rendered obvious over Ideses.

5. Claim 25

[25.0] The computer-implemented method according to claim 18,

202. See [18.0] - [18.6].

[25.1] wherein the machine learning model comprises a clustering model.

203. It is my opinion that Ideses discloses the additional limitations of dependent claim 25.

204. Ideses discloses the use of machine learning models. For example, "the learning module performs the machine learning and/or classifier training." Ex. 1005, ¶0095. I refer also to my analysis in section [18.6]. Ideses explicitly discloses "non-supervised approach is used in which labeling is based on clusterization. Optionally, clusters are generated automatically using a non-supervised and/or a semi-

supervised clustering software module" (clustering model). Ex. 1005, ¶0085. One skilled in the art would readily understand that the labelling based on clusterization and a semi-supervised clustering software module are machine learning implementing a clustering model. Clustering is a form of machine learning in which a set of feature vectors is typically split into a group of clusters. Each cluster consists of similar feature vectors. If, for instance, we have a set of malware feature vectors, a clustering algorithm might try to generate multiple clusters so that each cluster contains variants of the same malware. Understanding which feature vectors belong together (i.e., should be placed in the same cluster) and which ones do not (i.e., should not be placed together in the same cluster) is the goal of clustering algorithms.

205. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 25 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 25 is rendered obvious over Ideses.

6. Claim 26

[26.0] The computer-implemented method according to claim 18,

206. See [18.0] - [18.6].

[26.1] wherein the executable file is identified in response to a detected condition.

207. It is my opinion that Ideses discloses the additional limitations of dependent claim 26.

208. As of the priority date of the '869 patent, software applications could be installed on a mobile device in many ways, e.g., by downloading the application from the Internet or connecting the phone to a device (e.g. a laptop or desktop) and downloading from that device, or by a third party with remote access to the device, and a number of other ways. It was also common for the mobile device owner to be asked for permission prior to a new software being installed on his or her device.

209. In particular, Ideses discloses several conditions in which an executable file is identified. For example, "at 206, a software application is received at the mobile device, for example, the software application is downloaded from the internet, loaded using physical computer readable media, uploaded by a third party, or other methods of receipt. Optionally, the software application is requesting (automatically or manually) to be installed on the mobile device." Ex. 1005, ¶0131. Each of the respective receipt examples of receipt at the mobile device and software application request to be installed are detected conditions.

210. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 26 in view of the above-

described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 26 is rendered obvious over Ideses.

7. Claim 27

[27.0] The computer-implemented method according to claim 26,

211. See [26.0] - [26.1].

[27.1] wherein the detected condition is user request for a file download.

212. It is my opinion that Ideses discloses the additional limitations of dependent claim 27.

213. Ideses discloses "at 206, a software application is received at the mobile device, for example, the software application is downloaded from the internet, loaded using physical computer readable media, uploaded by a third party, or other methods of receipt." Ex. 1005, ¶0131. The phrase "software application is downloaded" (e.g., a file download) was commonly understood as of the priority date of '869 patent to be software downloaded from a website or a remote computer server.

214. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 27 in view of the above-

described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 27 is rendered obvious over Ideses.

8. Claim 28

[28.0] The computer-implemented method according to claim 26,

215. See [26.0] - [26.1].

[28.1] wherein the detected condition is the detection of a new file attempting to execute.

216. It is my opinion that Ideses discloses the additional limitations of dependent claim 28.

217. Ideses discloses "[t]he method of FIG. 2 may be automatically executed by software, for example, the mobile device automatically updates itself with the latest feature set, automatically detects software trying to install itself, automatically classifies the software, and/or automatically removes possibly harmful software." Ex. 1005, ¶0148. One of ordinary skill in the art would understand that the detected condition of a new file attempting to execute is software that automatically looks to see if a new file within a monitored system is trying to run or execute, as explained by Ideses.

218. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 28 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 28 is rendered obvious over Ideses.

9. Claim 29

[29.0] The computer-implemented method according to claim 18,

219. See [18.0] - [18.6].

[29.1] wherein the plurality of static data points represent predefined character strings in the executable file.

220. It is my opinion that Ideses discloses the additional limitations of dependent claim 29.

221. Ideses discloses broadly that "[t]he extracted features may be any feature that describes the software application." Ex. 1005, ¶0089. "For example, the extracted features may contain static meta data regarding the application, for example, icon, name, rating or other features." *Id.*

222. As I explained above in my discussion of character strings, one of ordinary skill in the art would have understood that character strings can be used for many

different purposes on a computer. Given Ideses further disclosure that "any feature that describes the software application," including the specific examples of "icon" and "name," which are commonly held in data structure like character strings, such examples would include predefined character strings in the executable file. Ex. 1005, ¶0089.

223. The use of predefined character strings for malware detection was well-known prior to the priority date of the '869 patent. As an example, Cheng et al. in 2013 stated, " . . . using pre-defined specific API calls and strings to train a malware classifier for each family using the SVM algorithm gives an average accuracy rate of 88%." Ex. 1032 at 1682.

224. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 29 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 29 is rendered obvious over Ideses.

10. Claim 30

[30.0] The computer-implemented method according to claim 18,

225. See [18.0] - [18.6].

[30.1] wherein a determination of whether the executable file is harmful is used to retrain the classifier.

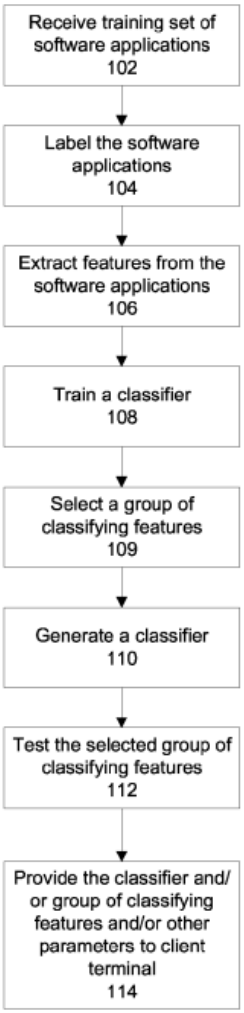
226. It is my opinion that Ideses discloses the additional limitations of dependent claim 30.

227. Limitation 30.1 first requires a determination of whether an executable file is harmful. As I have explained above, once a classifier has been trained to distinguish between two classes (e.g., malicious vs. benign software applications), a new software application can be classified by applying the trained classifier to the feature vector of any new (i.e., not previously seen during training) application. A process of checking whether the feature vector of the new software application satisfies or does not satisfy the learned condition can then be conducted. Depending on the outcome of this "check," the classifier will label the software application as either "malicious" or "benign."

228. Ideses discloses "at 214, the mobile device reports back to the server." Ex. 1005, ¶0144. "[T]he classification type outcome is reported." *Id.* Accordingly, the classification type outcome reporting is the checking of the software application against the feature vector, thus making a determination of whether the executable file is harmful.

229. Ideses further discloses "information regarding the software application is sent back as part of a **feedback loop**." Ex. 1005, ¶0144 (emphasis added). And

further, "the central server **learns about the existence of new software applications based on the feedback**." *Id.* at ¶0145 (emphasis added). "Referring back to FIG. 1, optionally, at 102, **multiple software applications for training the classifier are received**, for example, by central processor 302." *Id.* at ¶0080 (emphasis added). FIG. 1 of Ideses is copied below:



230. One of ordinary skill in the art would have understood that the central server learns about the existence of new software applications and their respective classification types as part of the feedback loop in order to retrain the classifier. In particular, one of ordinary skill in the art would understand this to mean that after some period has elapsed since training, a new set of apps is provided for additional training using the combination of the first training set with the new apps.

231. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 30 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 30 is rendered obvious over Ideses.

11. Claim 31

[31.0] The computer-implemented method according to claim 18

232. See [18.0] - [18.6].

[31.1] further comprising preventing execution of the executable file based on classifying the executable file as harmful.

233. It is my opinion that Ideses discloses the additional limitations of dependent claim 31.

234. Ideses discloses "at 212, a course of action is decided for the software application based on the automated classification." Ex. 1005, ¶0141. "[T]he software application is not installed, deleted, flagged, or otherwise prevented from functioning on the mobile device, for example, if the classification type is determined to be harmful, malware, bothersome, adware, intrusive, spyware, or other non-desirable classification types." *Id.* In particular, one of ordinary skill in the art would understand that if a file was classified to be harmful by an automated classification algorithm or system, then the software application associated with the file would be prevented from functioning using any one of several possible methods, as provided by the several examples in Ideses.

235. It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 31 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 31 is rendered obvious over Ideses.

12. Claim 32

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

236. I understand that a preamble typically does not limit a claim. However, to the extent that the preamble is limiting, it is my opinion that it is disclosed by Ideses.

237. Limitation [32.0] first requires a computer-readable storage device. Ideses explicitly discloses a non-exhaustive list of storage media which one of skill in the art would understand to be computer readable storage media in paragraphs [0063] and [0064]. Ex. 1005, ¶¶0063, 0064.

238. Limitation [32.0] also requires the storage device to contain instructions that cause a processor to execute. Ideses explicitly discloses that "[t]he computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon . . . The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device." Ex. 1005, ¶¶0063-0064. One of ordinary skill in the art would understand that an instruction execution device is, for example, a processor.

239. Limitation [32.0] also requires the processor to execute a process. Ideses discloses "[t]he present invention may be . . . a method." Ex. 1005, ¶0063. One of ordinary skill in the art would understand that a method is a process.

240. In one specific example, Ideses discloses a processor executing instructions as data modules, "[s]ystem 300 includes a central processor 302 with a memory 304 for storing data modules 306 thereon." Ex. 1005, ¶0073. Examples of execution of a process include, "[c]entral processor 302 may have sufficient processing ability to train a classifier based on a full set of features, for example, extracted from software applications." *Id.* "Modules 306 may include a feature extraction and/or pruning system." *Id.* at ¶0074. "Modules 314 may include a run-time feature extractor for extracting features from a software application for classification based on the selected group of classifying features." *Id.* at ¶0076. At the time of priority date of the '869 patent, one of ordinary skill in the art would have understood a system for classifier based detection of malware to include a computer (and hence a central processing unit) and a system to extract features and select relevant features from a body of training data during the training phase. Once the training phase is concluded, the same feature extraction module can be used at run-time to extract features from a new software application for which a benign vs. malicious prediction is desired.

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

241. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.1].

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

242. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.2].

[32.3] identifying an executable file to be evaluated;

243. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.3].

[32.4] extracting a plurality of static data points from the executable file;

244. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.4].

[32.5.a] generating a feature vector from the plurality of static data points

245. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.a].

[32.5.b] using a classifier trained to classify the static data points based on training data,

246. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.b].

[32.5.c] the training data comprising files known to fit into one of the plurality of categories of files,

247. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.c].

[32.5.d] wherein one or more features of the feature vector are selectively turned on or off

248. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.d].

[32.5.e] based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

249. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.e].

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

250. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.6].[1]

13. Claim 34

[34.0] The computer-readable storage device according to claim 32,

251. See [32.0] - [32.6].

[1] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 32 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 32 is rendered obvious over Ideses.

[34.1] wherein the machine learning model comprises an artificial neural network.

252. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [20.1].[2]

14. Claim 36

[36.0] The computer-readable storage device according to claim 32,

253. See [32.0] - [32.6].

[2] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 34 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 34 is rendered obvious over Ideses.

[36.1] wherein the machine learning model comprises a machine learning decision tree.

254. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [22.1].[3]

15. Claim 38

[38.0] The computer-readable storage device according to claim 32,

255. See [32.0] - [32.6].

[3] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 36 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 36 is rendered obvious over Ideses.

[38.1] wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

256. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [24.1].[4]

16. Claim 39

[39.0] The computer-readable storage device according to claim 32,

257. See [32.0] - [32.6].

[39.1] wherein the machine learning model comprises a clustering model.

258. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [25.1].[5]

---

[4] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 38 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 38 is rendered obvious over Ideses.

[5] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 39 in view of the above-described

17. Claim 40

[40.0] The computer-readable storage device according to claim 32,

259. See [32.0] - [32.6].

[40.1] wherein the executable file is identified in response to a detected condition.

260. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [26.1].[6]

---

teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 39 is rendered obvious over Ideses.

[6] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 40 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 40 is rendered obvious over Ideses.

18. Claim 41

[41.0] The computer-readable storage device according to claim 40,

261. See [40.0] - [40.1].

[41.1] wherein the detected condition is user request for a file download.

262. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [27.1].[7]

19. Claim 42

[42.0] The computer-readable storage device according to claim 40,

263. See [40.0] - [40.1].

---

[7] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 41 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 41 is rendered obvious over Ideses.

[42.1] wherein the detected condition is the detection of a new file attempting to execute.

264. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [28.1].[8]

20. Claim 43

[43.0] The computer-readable storage device according to claim 32,

265. See [32.0] - [32.6].

[8] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 42 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 42 is rendered obvious over Ideses.

[43.1] wherein the plurality of static data points represent predefined character strings in the executable file.

266. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [29.1].[9]

21. Claim 44

[44.0] The computer-readable storage device according to claim 32,

267. See [32.0] - [32.6].

[9] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 43 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 43 is rendered obvious over Ideses.

[44.1] wherein a determination of whether the executable file is harmful is used to retrain the classifier.

268. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [30.1].[10]

22. Claim 45

[45.0] The computer-readable storage device according to claim 32,

269. See [32.0] - [32.6].

[10] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 44 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 44 is rendered obvious over Ideses.

[45.1] wherein the process further comprises preventing execution of the executable file based on classifying the executable file as harmful.

270. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [31.1].[11]

23. Claim 46

[46.0] A system comprising:

271. I understand that a preamble typically does not limit a claim. However, to the extent that the preamble is limiting, it is my opinion that it is disclosed by Ideses.

272. For example, Ideses explicitly discloses a system. "System 300 includes a central processor 302 with a memory 304 for storing data modules 306 thereon." Ex. 1005, ¶0073. At the time of the priority date of the '869 patent, one of ordinary skill in the art would have understood that a system involving almost any computational

[11] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 45 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 45 is rendered obvious over Ideses.

task would include a Central Processing Unit or CPU, some memory (also called RAM or random access memory) and some storage (e.g., on a hard drive).

[46.1] at least one memory;

273. It is my opinion that Ideses discloses each part of this limitation.

274. For example, Ideses explicitly discloses memory 304 in the context of system 300, "[s]ystem 300 includes a central processor 302 with a memory 304 for storing data modules 306 thereon." Ex. 1005, ¶0073.

[46.2] at least one processor configured to perform operation of:

275. It is my opinion that Ideses discloses each part of this limitation.

276. For example, Ideses explicitly discloses processor 302 in the context of system 300, "[s]ystem 300 includes a central processor 302 with a memory 304 for storing data modules 306 thereon." Ex. 1005, ¶0073. Ideses further discloses that "computer readable program instructions may be provided to a processor." *Id.* at ¶0068.

277. Ideses further discloses that the instructions "execute via the processor." *Id.* One of ordinary skill in the art would have understood that a processor is configured to perform operation of the instructions. Moreover, one of ordinary skill in the art would have understood that computer readable program instructions are executed by

a computer which in turn has a processing unit or processor that actually executes the instructions.

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

278. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.1].

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

279. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.2].

[46.5] at least one processor configured to perform operation of:

280. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [46.2].

[46.6] identifying an executable file to be evaluated;

281. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.3].

[46.7] extracting a plurality of static data points from the executable file;

282. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.4].

[46.8.a] generating a feature vector from the plurality of static data points

283. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.a].

[46.8.b] using a classifier trained to classify the static data points based on training data,

284. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.b].

[46.8.c] the training data comprising files known to fit into one of the plurality of categories of files,

285. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.c].

[46.8.d] wherein one or more features of the feature vector are selectively turned on or off

286. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.d].

[46.8.e] based at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range; and

287. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.5.e].

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

288. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [18.6].[12]

24. Claim 48

[48.0] The system according to claim 46,

289. See [46.0] - [46.9].

[12] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 46 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 46 is rendered obvious over Ideses.

[48.1] wherein the machine learning model comprises an artificial neural network.

290. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [20.1].[13]

25. Claim 50

[50.0] The system according to claim 46,

291. See [46.0] - [46.9].

[13] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 48 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 48 is rendered obvious over Ideses.

[50.1] wherein the machine learning model comprises a machine learning decision tree.

292. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [22.1].[14]

26. Claim 52

[52.0] The system according to claim 46,

293. See [46.0] - [46.9].

[14] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 50 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 50 is rendered obvious over Ideses.

[52.1] wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

294. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [25.1].[15]

27. Claim 53

[53.0] The system according to claim 46,

295. See [46.0] - [46.9].

[53.1] wherein the machine learning model comprises a clustering model.

296. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [25.1].[16]

---

[15] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 52 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 52 is rendered obvious over Ideses.

[16] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 53 in view of the above-described

28. Claim 54

[54.0] The system according to claim 46,

297. See [46.0] - [46.9].

[54.1] wherein the executable file is identified in response to a detected condition.

298. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [26.1].[17]

---

teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 53 is rendered obvious over Ideses.

[17] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 54 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 54 is rendered obvious over Ideses.

29. Claim 55

[55.0] The system according to claim 54,

299. See [54.0] - [54.1].

[55.1] wherein the detected condition is user request for a file download.

300. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [27.1].[18]

30. Claim 56

[56.0] The system according to claim 54,

301. See [54.0] - [54.1].

[18] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 55 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 55 is rendered obvious over Ideses.

[56.1] wherein the detected condition is the detection of a new file attempting to execute.

302. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [28.1].[19]

31. Claim 57

[57.0] The system according to claim 46,

303. See [46.0] - [46.9].

[19] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 56 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 56 is rendered obvious over Ideses.

[57.1] wherein the plurality of static data points represent predefined character strings in the executable file.

304. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [29.1].[20]

32. Claim 58

[58.0] The system according to claim 46,

305. See [46.0] - [46.9].

[20] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 57 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 57 is rendered obvious over Ideses.

[58.1] wherein a determination of whether the executable file is harmful is used to retrain the classifier.

306. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [30.1].[21]

33. Claim 59

[59.0] The system according to claim 46, further comprising at least one processor configured to perform operation of:

307. See [46.0] - [46.9].

[21] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 58 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 58 is rendered obvious over Ideses.

[59.1] preventing execution of the executable file based on classifying the executable file as harmful.

308. It is my opinion that Ideses discloses each part of this limitation for the same reasons I expressed above with respect to claim limitation [31.1].[22]

B. Ground 2: Claims 18-20, 22, 24-34, 36, 38-48, 50, 52-59 are obvious under 35 U.S.C. § 103 over Ideses in view of Sharma

1. Reasons to Combine Ideses and Sharma

309. In my opinion, one of ordinary skill in the art would have been motivated to combine the teachings of Ideses and Sharma for multiple reasons. The obvious, beneficial, and predictable results of the combination include improved prediction performance and improved feature vector creation (by selecting highly relevant features from a potentially large set of features, some of which may have been irrelevant) which both enhance detecting and preventing the potentially harmful

[22] It is further my opinion that a POSITA would have had a reasonable expectation of successfully producing the invention of claim 59 in view of the above-described teachings of Ideses because the disclosure of Ideses coupled with the knowledge of one skilled in the art involves nothing more than the use of known components in accordance with their conventional uses in a predictable art. Thus, claim 59 is rendered obvious over Ideses.

effects of malware, as well as improved efficiency and reduced resource cost. Doing so would have been within the skillset of one of ordinary skill in the art and would produce beneficial, predictable results.

310. It is my opinion, that when considering the teachings of Ideses, a person of ordinary skill in the art would also have considered the teachings of Sharma. Both Ideses and Sharma are analogous prior art in the same field of endeavor as the '869 patent – detecting and preventing malware using learning classifiers. Ex. 1005, Abstract, ¶¶0023, 0058; Ex. 1006, Abstract, ¶¶0024-0026.

311. It is my opinion that a person of ordinary skill in the art would have been specifically motivated to incorporate several known features from Sharma. A POSITA would be motivated to incorporate Sharma's techniques for feature extraction, because doing so would address a known known problem and deficiency in Ideses, as will be discussed further below. A person of ordinary skill in the art would be motivated to incorporate Sharma's additional extracted features, because doing improves Ideses' feature selection and enhances Ideses' predictive performance, as will be discussed further below. Further, a POSITA would be motivated to incorporate Sharma's techniques for rules-based compliance checking, because doing so would further address an inefficiency in Ideses and further enhances Ideses' predictive performance, as will be discussed further below.

312. In my opinion, the combination of Ideses and Sharma would have simply combined prior art elements (techniques of feature extraction, including in a non-executing modality in Ideses; static extraction tools in Sharma) according to known devices (systems of training and applying a classifier) to yield predictable results (better-trained systems for training and applying a classifier) as Ideses intended, and thus would have been obvious to a person of ordinary skill in the art.

313. More particularly, a person of ordinary skill in the art would have been specifically motivated to incorporate Sharma's known techniques for feature extraction because doing so would address a known problem and deficiency in Ideses.

314. Ideses discloses that "[a]n aspect of some embodiments of the present invention relates to systems and methods for selecting a set of classifying features extractable from Software applications." Ex. 1005 ¶0044. However, Ideses does not provide particular methods for extracting said features. Ideses does recognize that the method and cost of extracting features is an important consideration when determining which features to select for use and hence which features to extract. Ideses discloses that "[t]he selected features are significant to the classification process." Ex. 1005 ¶0045. Ideses further discloses that "the classification features are selected based on an identified resource requirements for extracting each of the features. For example, the memory required to extract the feature, the processor

usage required to extract the feature, the time to extract the feature, and/or other factors." *Id.* at ¶0046.

315. Sharma discloses a host of methods for extracting features from applications. Ex. 1006, ¶¶0040-0045. Specifically, Sharma discloses that "[t]he RRF system uses features of a computer application to evaluate its risk. The features of a computer application can be extracted using a variety of tools that extract static, dynamic, and/or manifest features of a computer application. . . . The tools may be a software application that is available for use online or via a machine-readable storage-medium. The tools may also be modified to suit the purposes of the RRF system, for example, the tools may be modified to resolve inconsistencies in their output, and to parallelize them for efficiency." *Id.* at ¶0040. Sharma further discloses examples of tools in paragraphs 0041-0045.

316. A person of ordinary skill in the art would be motivated to combine Sharma's methods of feature extraction to overcome Ideses' lack of feature extraction methods. Specifically, a person of skill in the art would be compelled to incorporate Sharma's disclosed use of tools to perform feature extraction to improve Ideses ability to extract features that may have otherwise been resource prohibitive, which enhances Ideses predictive performance.

317. Incorporating Sharma's features with Ideses' system would have not only been advantageous, but a person of skill in the art would have had a reasonable

expectation of success in combining said functionality as the combination involves the use of known techniques and components in accordance with their conventional uses in a predictable art.

318. Additionally, a person of skill in the art would have been specifically motivated to incorporate Sharma's disclosed features in training Ideses' machine learning model in order to enhance Ideses' predictive performance.

319. Ideses suggests a variety of features that may be extracted from a file for training a classifier. Ideses suggests extracting features that "may contain static meta data regarding the application, for example, icon, name, rating, or other features. In another example, the extracted features may contain information regarding the executable code and/or software package, for example, in the form of byte code, resources, permissions, or other features. In yet another example, another modality of features may include behavioral features, for example, temporal information regarding system calls, system usage, CPU usage, network utilization, or other features. The features may include suitable data and/or meta data that may be extracted from the software application and/or computed. Examples of extracted features include: application name, icon, rating, permissions, internal function calls, decompiled byte code, behavioral properties such as network, CPU, user interface (UI), and/or systems call usage, and/or other suitable quantifiable measures that may be obtained for the software application." Ex. 1005, ¶0089. Ideses further suggests

that "[t]he extracted features may be any feature that describes the software application. The extracted features may be varied, containing information from different modalities." Ex. 1005, ¶0089.

320. Sharma proposes a number of features not considered by Ideses such as manifest-related features and risk score features. For example, Sharma proposes that "[t]he classifiers use manifest and static features computationally extracted from the mobile computer application to provide a risk-rating in a selected range . . . . The risk assessment is based on manifest and static features extracted from the computer applications using one or more feature extraction tools." Ex. 1006, ¶0024. Sharma further discloses that "[a] manifest analysis tool may extract features and information from an applications' manifest file. The manifest file contains information regarding files that are included in the computer application package." *Id.* at ¶0027. Sharma specifically discloses, that "some of the features that may be extracted and analyzed include, but are not limited to, declared permissions, declared intents, declared hardware features, used permissions, used intents, used hardware features (such as GPS, memory, camera), code execution, audio/video eavesdropping, certain function or API calls, device settings harvesting, connection interfaces exfiltration, Suspicious connection establishment, telephony services abuse, telephony identifiers leakage, location lookup, and personal information management (PIM) data leakage," which are not features mentioned in Ideses. *Id.*

321. Sharma further discloses that "[a]nalysis of such features can help identify computer applications that exhibit malware or malicious characteristics. For example, when a computer application has a set of declared permissions, intents, and hardware features, but only uses a subset of the declared permissions, intents and hardware features, such characteristics may contribute to the computer application's risk rating." Ex. 1006 ¶0027. It is my opinion that a person of ordinary skill in the art would have been motivated to combine the features disclosed in Sharma with Ideses' system in order to enhance the predictive performance of Ideses.

322. It was well-known by the priority date of the '869 patent that adding additional features to a machine learning classifier often led to improved predictive performance of the classifier. When an initial training set is built it includes a set of features (e.g., those features defined by Ideses as discussed above). However, there may be other factors (or features) that play a key role in distinguishing malicious applications from benign ones (i.e., factors that were missed and/or not disclosed by Ideses). As such, incorporation of those features taught by Sharma that were not considered by Ideses into Ideses' system would allow for an evaluation of an improvement in predictive quality of the classifier.

323. It was well known that new, different, and/or additional features enhance the predictive quality of a machine learning classifier. As an example, in my own work on predicting whether accounts on Twitter are bots or not, my co-authors and I

incorporated features provided by prior work in the field that we may have not originally contemplated. *See* Ex. 1029 (incorporating features identified in Ex. 1033). My colleagues and I were motivated to include these features in the training set for our machine learning model to enhance the predictive quality of the machine learning model.

324. Incorporating Sharma's features with Ideses' system would have not only been advantageous, but a person of skill in the art would have had a reasonable expectation of success in combining said functionality as the combination involves the use of known techniques and components in accordance with their conventional uses in a predictable art.

325. Additionally, a person of skill in the art would have been specifically motivated to incorporate Sharma's known techniques for rule-based compliance checking and incorporation of a threat model to assist in identifying harmful and benign applications because doing so would improve Ideses' predictive performance, improve Ideses' efficiency, and expand the number of applications Ideses can predict.

326. Sharma discloses that "[t]he extracted features are used by the classifiers 70 to analyze computer applications 55. The classifiers 70 include Bayesian classifiers, Partial Least Square (PLS) classifiers, Compliance Checking algorithm, and Rule-based algorithm . . . . The Compliance Checking and Rule-based algorithms

determine if a user-defined security guideline is violated. These algorithms may use rules and/or heuristic methods to determine violations of user-defined security guidelines. For example, based on the permissions of the application and API calls the application makes, a determination is made whether the application is writing to a persistent memory, such as a Secure Digital (SD) memory card, or whether the application executes a UNIX command." Ex. 1006 ¶0028.

327. Sharma discloses that, "[i]n some embodiments, the computer application analyzed is an application run on Android OS or an application run on iOS. In an example embodiment, the features of the Android application are extracted and analyzed using the PLS classifier, or the Bayesian classifier, or the PLS and the Bayesian classifier. In another example embodiment, the features of the iOS application are analyzed by the compliance checking algorithm and the rule-based algorithm (discussed above). The iOS system is a closed system, and not many tools are available to extract features from an iOS application (compared to the tools available for feature extraction for an Android application). Therefore, a subset of features are analyzed for the iOS application (versus the larger set of features analyzed for an Android application). The subset of features for an iOS application may be analyzed by the rule-based algorithm or may be analyzed in a similar manner to the features in an Android application (via PLS classifier and/or Bayesian classifier)." Ex. 1006 ¶0083.

328. As discussed above, Ideses discloses that "[a]n aspect of some embodiments of the present invention relates to systems and methods for selecting a set of classifying features extractable from Software applications, in order to generate classifiers, provide the selected group of classifying features (e.g., feature vectors), select coefficients and/or other parameters for classification of Software applications." Ex. 1005 ¶0044. Ideses further discloses that "[t]he extracted features may be any feature that describes the software application. The extracted features may be varied, containing information from different modalities. For example, the extracted features may contain static metadata regarding the application, for example, icon, name, rating or other features. In another example, the extracted features may contain information regarding the executable code and/or software package, for example, in the form of byte code, resources, permissions, or other features. In yet another example, another modality of features may include behavioral features, for example, temporal information regarding system calls, system usage, CPU usage, network utilization, or other features. The features may include suitable data and/or meta data that may be extracted from the Software application and/or computed. Examples of extracted features include: application name, icon, rating, permissions, internal function calls, decompiled byte code, behavioral properties such as network, CPU, user interface (UI), and/or system calls

usage, and/or other suitable quantifiable measures that may be obtained for the software application." *Id.* at ¶0089.

329. A person of skill in the art would have been motivated to combine Sharma's rules-based compliance checking and threat model with Ideses' training and classification based on feature extraction for several reasons. As indicated in Sharma, some applications are difficult to extract features from and/or some applications have limited sets of features. *See e.g.*, Ex. 1006 ¶83 ("The iOS system is a closed system, and not many tools are available to extract features from an iOS application."). Sharma discloses evaluating such an application with rules-based compliance checking algorithms. Ideses further recognizes the difficulty in extracting features from an application. For example, Ideses discloses "the classification features are selected based on an identified resource requirements for extracting each of the features. For example, the memory required to extract the feature, the processor usage required to extract the feature, the time to extract the feature, and/or other factors." Ex. 1005 ¶0046. Accordingly, a person of skill in the art would be motivated to incorporate Sharma's rules-based compliance checking algorithms with Ideses' machine learning classifier in order to increase the system's predictive functionality and run-time efficiency for applications with low quantities of extractable features and for applications that would require significant resource requirements to extract the features.

330. One of ordinary skill in the art would understand that the rules-based compliance checking algorithms disclosed in Sharma could be combined with Ideses' system in several ways. One of ordinary skill in the art would understand that the rules-based compliance checking algorithms disclosed in Sharma could be used in conjunction with the features-based machine learning classifier disclosed in Ideses for applications with limited extractable features because doing so would be likely to increase Ideses' predictive performance.

331. Additionally, one of ordinary skill in the art would also understand that the rules-based compliance checking algorithms disclosed in Sharma could be used to generate additional features for inclusion in the features-based machine learning classifier disclosed in Ideses. Sharma discloses that "[t]he Compliance Checking and Rule-based algorithms determine if a user-defined security guideline is violated. These algorithms may use rules and/or heuristic methods to determine violations of user-defined security guidelines." Ex 1006, ¶0028. A person of ordinary skill in the art would have been motivated to use this teaching of Sharma to build a rule-based tool that can flag whether a user-defined security guideline has been violated or not in Ideses. It would have also been obvious to a person of ordinary skill in the art that the output of a tool that checks for violations of user-defined security guidelines would have been valuable additional input features to Ideses' machine learning and feature selected models. For instance, it would have been obvious to any person of

ordinary skill in the art at the time that if the rule-based algorithm flagged a violation of a user-defined security guideline by an app, that could be an important method to add a new feature with value 0 (no observed violation of user-defined security guideline) or 1 (violation flagged of a user-defined security guideline). In fact, it would have been obvious to a person of ordinary skill in the art that if the rule-based algorithm looked for potential violations of multiple rules, each of those rules could be used to generate a feature of the kind above. It would have been obvious to a person of ordinary skill in the art that an application that violates known security guidelines could potentially be malicious and hence, a feature or set of features based on such user-defined security guidelines would be of immediate and obvious interest for including in the set of features used to determine the classification of an application.

332. As discussed above, it was well known that new, different, and/or additional features enhance the predictive quality of a machine learning classifier. Further, as indicated in Ideses it was well known that efficiency in classification and extraction of features was important. Accordingly, one of skill in the art would be motivated to incorporate the rules-based compliance checking classifiers of Sharma with Ideses' system.

333. Incorporating Sharma's rules-based compliance checking classifiers with Ideses' system would have not only been advantageous, but a person of skill in the

art would have had a reasonable expectation of success combining said functionality as the combination involves the use of known techniques and components in accordance with their conventional uses in a predictable art.

334. Accordingly, the expectation of success in combining Ideses with Sharma is evident because the combination of Ideses and Sharma involved the use of known techniques and components in accordance with their conventional uses in a predictable art.

2. Claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, 52-59

335. See Section VIII.A. above. All statements are hereby incorporated herein and will not be repeated.[23]

3. Elements [18.5.b], [32.5.b], and [46.5.b] (as recited above in Section VIII.A.)

336. It is my opinion that Ideses discloses each of these limitations. It is further my opinion that Sharma discloses each of these limitations alone or in combination with Ideses.

---

[23] The reasonable expectation of success stated in Section VIII.B.1. above and Section VIII.B.3 below is applied to claims 18, 20, 22, 24-32, 34, 36, 38-46, 48, 50, and 52-59 as incorporated in Ground 2.

337. Referring to element [18.5.b], the limitation requires generating a feature vector "using a classifier trained to classify the static data points based on training data."

338. Sharma discloses that a classifier is trained based on training data. Sharma discloses that "[t]he classifiers are first trained and tested using an input data set that includes known-malware and known-benign computer applications." Ex. 1006, ¶0025. One of ordinary skill in the art would understand that the input data set described in Sharma is the recited training data of the '869 patent because Sharma explicitly discloses that classifiers are trained using an input data set having known categorizations, as I have previously explained; here, an input data set that includes known-malware and known-benign computer applications.

339. Next, Sharma discloses that a feature vector is generated using a classifier. For example, Sharma teaches "a feature vector is constructed using a Bayesian model classifier." Ex. 1006, ¶0047. One of ordinary skill in the art would understand that a Bayesian model classifier is an example of a classifier for constructing (e.g. generating) the feature vector.

340. Next, Sharma discloses that the classifier is trained to classify features such as static data points. For example, Sharma teaches "[f]or the Bayesian classifier, all the features are inputted to the classifier. The classifier analyses each feature and determines if each feature contributes to the probability that the computer

application is malware. The classifier is designed to consider each feature to contribute independently to the probability of being malware." Ex. 1006, ¶0047. As I have previously explained with respect to my analysis of [18.1], extracted features can include static data such as static data points. Sharma likewise explains that extracted features can include static data such as static data points. For example, "[a] static analysis tool may extract features and information regarding an applications' software code, such as, code size, run time, and the like." Ex. 1006, ¶0027. One of ordinary skill in the art would therefore understand that the classifier's consideration of each feature includes static data points.

341. Moreover, Sharma explicitly discloses consideration by a classifier of the independent contribution of each feature to a probability of being malware. One of ordinary skill in the art would understand that in order to consider the independent contribution of each feature to a probability of malware, the classifier is necessarily trained to classify the feature as part of the training of the classifier. Sharma therefore discloses a classifier trained to classify the static data points.

342. One of ordinary skill in the art would have had a reasonable expectation of success in producing the method recited in claim 18 through the combination of Ideses with Sharma. In particular, the combination is nothing more than applying Ideses' known techniques of generating a feature vector using a classifier to Sharma's known system that utilizes a classifier with particular consideration of

each feature to the contribution of a probability of malware as Ideses intended, and thus would have been obvious to one of ordinary skill in the art.

4. Claim 19

[19.0] The computer-implemented method according to claim 18,

343. See [18.0] - [18.6].

[19.1] wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

344. It is my opinion that Ideses alone or combined with Sharma renders this limitation obvious.

345. As I have explained, static data points are discrete points of data within a file that are obtained through static analysis, rather than dynamic analysis. As I have further explained, static analysis typically refers to the process of examining code as it stands (not run or executed).

346. Moreover, as I have also explained, encryption is the process of converting information/data into a code generally to obscure the information's meaning. Decryption is the process of reversing the encryption of information/data such that decryption algorithms convert an encrypted file back into its original form. Packing is the process of encrypting and compressing data. Unpacking is reversing the packing process.

347. Accordingly, in the terms of the '869 patent, static analysis of the static data points includes determination of characteristics of a program that can be obtained without running or executing the program code, including without reversing encryption of the program or reversing encryption and compression of the program.

348. Ideses explains that "extracted features may be varied, containing information from different modalities." Ex. 1005, ¶0089. Ideses specifically contrasts run-time feature extraction, which necessarily requires decrypting or unpacking the executable file, with off-line feature extraction. "The feature extraction may take place off-line, not part of a run-time operation." Ex. 1005, ¶0090. Ideses further explains that "[e]xtraction of the full set of features may not be possible during run-time on the mobile device. Extraction of the full set of features may take place at the central processor, independently of run-time operation of the mobile device." *Id.* at ¶0092.

349. Ideses further provides a number of examples of static data points that are extracted without decrypting or unpacking the executable file. As I have explained above, these static data points are otherwise referred to as static features, or features associated with a software program that can be obtained through a static analysis of the software program's code.

350. Ideses discloses, "[f]or example, the extracted features may contain static meta data regarding the application, for example, icon, name, rating or other

features. In another example, the extracted features may contain information regarding the executable code and/or software package." Ex. 1005, ¶0089. As I have explained, icon, name, and rating are examples of static data points (static features).

351. To the extent Ideses does not explicitly disclose static data point extraction without decrypting or unpacking the executable file, which is not conceded, Sharma renders this limitation obvious.

352. In particular, Sharma specifically discloses, referring to FIG. 1B, "tools 65 are programmed to extract certain features from the computer applications 55 for analysis. The extracted features are used by the classifiers 70 to evaluate the risk of computer applications 55." Ex. 1006, ¶0027. FIG. 1B of Sharma is copied below:

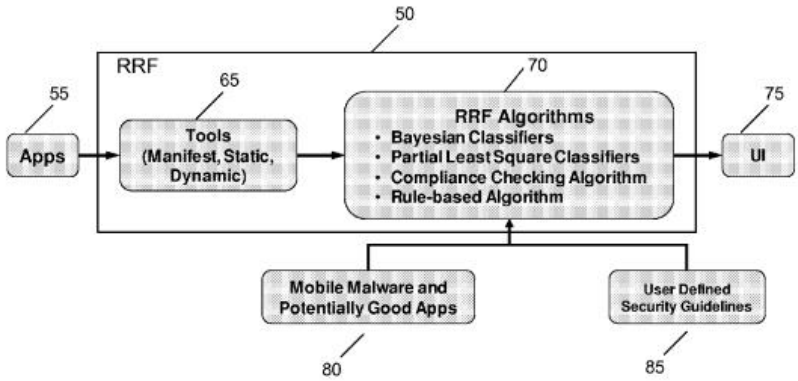


FIG. 1B

353. The static tool (65) described in Sharma is an example of a tool that can conduct static data point extraction without decrypting or unpacking the executable file, as I will further explain.

354. Sharma makes comparisons among manifest, static, and dynamic extraction tools. Sharma states, a "static analysis tool may extract features and information regarding an applications' software code, such as, code size, run time, and the like." Ex. 1006, ¶0027.

355. Sharma specifically contrasts its static extraction from its dynamic analysis tool. The dynamic analysis tool "extract[s] features and information related to the computer application when it is running or operating," which necessarily requires decrypting or unpacking the executable file. Ex. 1006, ¶0027. Thus, as I have explained, static tool 65 can extract some features without running or executing the program code.

356. It is my opinion that it would have been obvious to one of ordinary skill in the art to combine Ideses and Sharma to extract some static data points without decrypting or unpacking the executable file. Ideses already discloses that "extracted features may be any feature that describes the software application. The extracted features may be varied, containing information from different modalities." Ex. 1005, ¶0089.

357. For example, it would have been obvious to one of ordinary skill in the art at the priority date of the '869 patent to include binary features such as "packed" (set to 1 if the software application was packed, set to 0 otherwise) and "encrypted" (set to 1 if the software application was encrypted, set to 0 otherwise), as part of the

definition of features, which could be determined as part of a static analysis in Sharma. In fact, it was also known before the time of the '869 patent that it was possible to determine which packing algorithm was used to pack a software application. *See* Ex. 1035 ("Packer classification is an important step in malware analysis, if it can be designed so that it gives a reliable approach to detect packed files. The objective of packer classification is to quickly detect and identify the packer, allowing AV researchers to easily and correctly unpack the file and retrieve original payload for further malware detection and analysis." *Id.* at 371.); Ex. 1036 ("A packer is basically a program that produces a number of data blocks forming a compressed or encrypted version of the original executable . . . We developed a holistic method for identifying packing algorithms of given executables, no matter whether the packer of the executable is already known or not." *Id.* at 19, 20.); Ex. 1037 ("PE-Probe functions in two phases: (1) it classifies a given executable as packed or non-packed by employing well-studied heuristics, and (2) it invokes specialized structural models – separately developed for packed and non-packed executables – for malware detection on the basis of the outcome of the previous step." *Id.* at 1).

358. One of ordinary skill in the art would have had a reasonable expectation of success in producing the method recited in claim 19 through the combination of Ideses with Sharma. In particular, as I have explained, Ideses discloses known

techniques of feature extraction, including static data points in a non-executing modality. Sharma discloses known additional static and manifest feature extraction tools. The combination would yield the predictable results of a system for training and applying a classifier based on "any feature that describes the software application" as Ideses intended. Ex. 1005, ¶0089; Ex. 1006, ¶0025.

5. Claim 33

[33.0] The computer-readable storage device according to claim 32,

359. See [32.0] - [32.6].

[33.1] wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

360. It is my opinion that Ideses alone or combined with Sharma renders this limitation obvious for the same reasons I expressed above with respect to claim limitation [19.1].[24]

[24] It is my opinion one of ordinary skill in the art would have had a reasonable expectation of success in producing the method recited in claim 33 through the combination of Ideses with Sharma. In particular, as I have explained, Ideses discloses known techniques of feature extraction, including static data points in a non-executing modality. Sharma discloses known additional static and manifest

6. Claim 47

[47.0] The system according to claim 46,

361. See [46.0] - [46.9].

[47.1] wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

362. It is my opinion that Ideses alone or combined with Sharma renders this limitation obvious for the same reasons I expressed above with respect to claim limitation [19.1].[25]

---

feature extraction tools. The combination would yield the predictable results of a system for training and applying a classifier based on "any feature that describes the software application" as Ideses intended. Ex. 1005, ¶0089; Ex. 1006, ¶0025.

[25] It is my opinion one of ordinary skill in the art would have had a reasonable expectation of success in producing the method recited in claim 47 through the combination of Ideses with Sharma. In particular, as I have explained, Ideses discloses known techniques of feature extraction, including static data points in a non-executing modality. Sharma discloses known additional static and manifest feature extraction tools. The combination would yield the predictable results of a

IX. NO OBJECTIVE INDICIA OF NONOBVIOUSNESS

363. I am not aware of any objective indicia of non-obviousness related specifically to the ’869 patent. Indeed, I am not aware of any evidence of commercial success, long felt but unsolved need, failure of others, industry praise, or unexpected results specifically related to the ’869 patent.

X. CONCLUSION

364. For the reasons set forth above, it is my opinion that claims 18-20, 22, 24-34, 36, 38-48, 50, 52-59 of the ’869 patent are invalid in view of the prior art. In signing this declaration, I understand that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I acknowledge that I may be subject to cross-examination in this case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

365. I declare that all statements made herein of my knowledge are true, that all statements made on information and belief are believed to be true, and that these

---

system for training and applying a classifier based on “any feature that describes the software application” as Ideses intended. Ex. 1005, ¶0089; Ex. 1006, ¶0025.

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 1007 of Title 18 of the United States Code.

Executed: June 29, 2023

By:

/Venkatramanan Siva Subrahmanian/

Venkatramanan Siva Subrahmanian