

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

SOPHOS LTD. and SOPHOS INC.,
Petitioners

v.

WEBROOT INC.,
Patent Owner.

Case No. PGR2023-0031
U.S. Patent No. 11,409,869

Petition for Post-Grant Review of U.S. Patent No. 11,409,869

Mail Stop **PATENT BOARD**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

TABLE OF CONTENTS	i
TABLE OF EXHIBITS	iv
TABLE OF ABBREVIATIONS	vi
I. INTRODUCTION	1
II. MANDATORY NOTICES	2
A. Real Parties-in-Interest 37 C.F.R. § 42.8(b)(1)	2
B. Related Matters Under 37 C.F.R. § 42.8(b)(2)	3
1. Current Litigation.....	3
2. Administrative Proceedings	3
C. Lead and Back-Up Counsel.....	3
D. Service of Information Under 37 C.F.R. § 42.8(b)(4)	5
III. PAYMENT OF FEES	5
IV. REQUIREMENTS OF POST GRANT REVIEW	5
A. Standing.....	5
B. Identification of Challenge and Relief Requested	5
1. Ideses (Ex. 1005).....	7
2. Mori (Ex. 1006).....	7
C. Construction of the Challenged Claims Under 37 C.F.R. § 42.204(b)(3).....	7
D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.204(b)(4).....	9
E. Supporting Evidence Under 37 C.F.R. § 42.204(b)(5)	10
V. THRESHOLD REQUIREMENT FOR PGR	10
VI. PERSON OF ORDINARY SKILL IN THE ART	10
VII. STATE OF THE ART	11
1. Malware	11
2. Static and Dynamic Malware Analysis.....	11
VIII. OVERVIEW OF THE '869 PATENT	11

A.	Summary of the Prosecution History of the '869 Patent	12
IX.	DETAILED CHALLENGE	15
A.	Ground 1: Claims 1-59 Are Rendered Obvious By Ideses	15
1.	Prior Art Overview.....	15
2.	Analysis.....	16
B.	Ground 2: Claims 1-6, 8-24, 26-27, 29-30, 32-38, 40-41, 43-44, 46-52, 54-55, and 57-58 Are Rendered Obvious By Mori	59
1.	Prior Art Overview.....	59
2.	Analysis.....	61
C.	Ground 3: Claims 1-59 Are Rendered Obvious Over Ideses in View of Mori	83
1.	Overview of Ground	83
2.	Analysis of Unpatentability and Motivation to Combine.....	83
D.	Ground 4: Claims 1-59 Are Indefinite Under 35 U.S.C. § 112(b)	86
1.	Legal Standard	87
2.	The Independent Claims Have Unclear Antecedent Basis	87
E.	Grounds 5 and 6: Claims 8, 14, 17, 29, 43, and 57 Are Indefinite Under 35 U.S.C. § 112(b) and Lack Written Description	88
X.	PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION.....	94
A.	PTAB Should Not Exercise Its Discretion to Deny Institution Under <i>Fintiv</i>	94
1.	Factor 1: Likelihood of Stay	95
2.	Factor 2: District Court Schedule	95
3.	Factor 3: Petitioners' Investment in PGR Outweighs Forced Investment in Litigation to Date	97
4.	Factor 4: The Petition Raises Unique Issues	98
5.	Factor 5: Institution Will Provide the Board an Opportunity to Invalidate Claims That Later Could Be Reasserted Against Others	99
6.	Factor 6: Other Circumstances Support Institution	99

B.	PTAB Should Not Exercise its Discretion To Deny Institution Under <i>Becton</i> and <i>Advanced Bionics</i>	100
XI.	CONCLUSION.....	101
	APPENDIX: CHALLENGED CLAIM LISTING	104

TABLE OF EXHIBITS

Exhibit No.	Document
1001	U.S. Patent No. 11,409,869 (“the ’869 Patent”)
1002	File History for U.S. Patent No. 11,409,869
1003	File History for U.S. Patent No. 10,599,844
1004	Declaration of Dr. Gene Tsudik
1005	U.S. Patent Appl. Pub. No. 2015/0213376 (“Ideses”)
1006	Japanese Patent Appl. Pub. No. 2012027710A (“Mori”) and Certified Translation Thereof
1007	Litigation Claim Construction Order (D.I. 236)
1008	Litigation Opening Claim Construction Brief (D.I. 263)
1009	Litigation Responsive Claim Construction Brief (D.I. 304)
1010	Litigation Joint Stipulation Reserving Appellate Rights (D.I. 260)
1011	Monnappa K A, <i>Learning Malware Analysis</i> (2018)
1012	Michael Sikorski & Andrew Honig, <i>Practical Malware Analysis</i> (2012)
1013	John C. Platt, <i>Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods</i> , in <i>Advances in Large Margin Classifiers</i> (Alexander J. Smola <i>et al.</i> , eds. 2000)
1014	Wang <i>et al.</i> , <i>Detecting Unknown Malicious Executables Using Portable Executable Headers</i> , Fifth International Joint Conference on INC, IMS and IDC, 278-84 (2009)
1015	<i>Virus Bulletin</i> (Nov. 1990)
1016	U.S. Patent No. 8,709,924
1017	U.S. Patent No. 9,465,940
1018	D. Devi & S. Nandi, <i>Detection of Packed Malware</i> , in Proceedings of the First International Conference on Security of Internet of Things (SecurIT ’12), Association for Computing Machinery, New York, NY, USA, 22-26 (2012)

*Petition for Post-Grant Review
of U.S. Patent No. 11,409,869*

Exhibit No.	Document
1019	Excerpt from Transcript of Motions Hearing, <i>Webroot Inc. et al. v. AO Kaspersky Lab et al.</i> , No. 6:22-cv-00243 (W.D. Tex. Dec. 9, 2022)
1020	Shabtai <i>et al.</i> , <i>Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey</i> , Information Security Technical Report, 16-29 (2009)
1021	U.S. Patent Appl. Pub. No. 2010/0293273
1022	U.S. Patent Appl. Pub. No. 2010/0082642
1023	Defendant Sophos Ltd.'s Stipulation Regarding Invalidity Contentions for U.S. Patent No. 11,409,869

TABLE OF ABBREVIATIONS

Abbreviation	Term
'869 Patent	U.S. Patent No. 11,409,869 (Ex. 1001)
'844 Patent	U.S. Patent No. 10,599,844
AIA	America Invents Act
Board	Patent Trial and Appeal Board
Challenged Claims	Claims 1-59 of U.S. Patent No. 11,409,869
Ex.	Exhibit
Fig.	Figure
Ideses	U.S. Patent Appl. Pub. No. 2015/0213376 (Ex. 1005)
Litigation	<i>Webroot, Inc. v. AO Kaspersky Lab</i> , No. 6:22-cv-00243-ADA-DTG (W.D. Tex.), in which the following cases are consolidated: <i>Webroot Inc, et al. v. Sophos Ltd.</i> , No. 6:22-cv-240 (W.D. Tex.); <i>Webroot, Inc. et al. v. Trend Micro, Inc.</i> , No. 6:22-cv-239; <i>Webroot, Inc. et al. v. CrowdStrike Holdings, Inc. et al.</i> , No. 6:22-cv-241; <i>Webroot, Inc. et al. v. AO Kaspersky Lab</i> , No. 6:22-cv-243; <i>Webroot, Inc. et al. v. Forcepoint LLC</i> , 6:22-cv-00342
PGR	Post-Grant Review
Mori	Japanese Patent Appl. Pub. No. 2012027710A (Ex. 1006)
Office	U.S. Patent and Trademark Office
Patent Owner	Webroot, Inc.
Petitioners	Sophos Ltd. and Sophos Inc.
POSITA	Person of Ordinary Skill in the Art

I. INTRODUCTION

Pursuant to 35 U.S.C. § 321, Petitioners submit this Petition for PGR of the '869 Patent (Ex. 1001) and seek cancellation of claims 1-59.

The '869 Patent relates to “threat detection of executable files”—specifically, through a method whereby a “plurality of static data points” are extracted from an executable file. Ex. 1001, Abstract. The extracted static data points are analyzed by “applying a classifier.” *Id.* Based on the results of the analysis, a determination is made about whether the executable file is harmful. *Id.*

Every one of the claimed features of this allegedly novel malware detection process was disclosed in (or would have been obvious in light of) U.S. Patent Appl. Pub. No. 2015/0213376 (“Ideses”) (Ex. 1005), which was not considered by the Examiner during the '869 Patent’s prosecution and is a focus of this Petition. Ideses explicitly discloses the extraction of “static meta data”—*i.e.*, “static data points” as required by the '869 Patent’s claims—and the training and application of a classifier to detect malware. Ex. 1005, Abstract, [0089], [0134]-[0135]. Ideses also discloses “selectively” turning features of a classifier on or off based on whether a value of “one or more features of the feature vector are selectively turned on or off 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.” *E.g., id.*, [0104]-[0110].

The claimed features in the independent claims of the '869 Patent were also disclosed in (or would have been obvious in light of) Japanese Patent Appl. Pub. No. 2012027710A ("Mori") (Ex. 1006), which was likewise not considered by the Examiner. Like Ideses, Mori discloses a malware detection method that "focuses on readable strings contained in software files, collects information (characteristic vectors) that characterizes malware, such as messages contained in programs that constitute malware, host names, execution command names, function names and library names used for communication," and "compares this content with strings contained in files being inspected to detect malware." Ex. 1006, [0023].

The claims of the '869 Patent are also indefinite under 112(b) and/or lack written description, for the reasons discussed below.

This Petition demonstrates that there is a reasonable likelihood that Petitioners will prevail on at least one of the Challenged Claims. Therefore, PGR should be instituted and claims 1-59 should be canceled as unpatentable under §§ 102, 103, and 112.

II. MANDATORY NOTICES

A. Real Parties-in-Interest 37 C.F.R. § 42.8(b)(1)

Sophos Ltd., and Sophos, Inc. are the real parties-in-interest.

B. Related Matters Under 37 C.F.R. § 42.8(b)(2)

1. Current Litigation

The '869 Patent and its parent patent, the '844 Patent, are presently asserted in five suits brought by Patent Owner against Sophos Ltd. and four other defendants: *Webroot Inc, et al. v. Sophos Ltd.*, No. 6:22-cv-240 (W.D. Tex.); *Webroot, Inc. et al. v. Trend Micro, Inc.*, No. 6:22-cv-239; *Webroot, Inc. et al. v. CrowdStrike Holdings, Inc. et al.*, No. 6:22-cv-241; *Webroot, Inc. et al. v. AO Kaspersky Lab*, No. 6:22-cv-243; *Webroot, Inc. et al. v. Forcepoint LLC*, 6:22-cv-00342. These cases are consolidated as *Webroot, Inc. v. AO Kaspersky Lab*, No. 6:22-cv-00243-ADA-DTG (W.D. Tex.) (“the Litigation”).

2. Administrative Proceedings

Petitioners are not aware of any proceedings involving the '869 Patent. The '844 Patent ('869 Patent's parent) is the subject of the following IPR petitions: *Sophos Inc. et. al. v. Open Text Inc. et al.*, in IPR2023-00528 and *Trend Micro, Inc. v. Webroot, Inc. et al.* in IPR2023-00662.

C. Lead and Back-Up Counsel

Petitioners provide the following designation of counsel:

*Petition for Post-Grant Review
of U.S. Patent No. 11,409,869*

Lead Counsel	Backup Counsel
Benjamin Hershkowitz (Reg. No. 41,833) 200 Park Avenue New York, NY 10166 Telephone: 212.351.2410 Facsimile: 212.351.6210 bhershkowitz@gibsondunn.com	Josh Krevitt (<i>pro hac vice</i> motion requested) Gibson, Dunn & Crutcher LLP 1881 Page Mill Road Palo Alto, CA 94304-1211 Tel: (212) 351-2490 JKrevitt@gibsondunn.com
	Brian A. Rosenthal (<i>pro hac vice</i> motion requested) Gibson, Dunn & Crutcher LLP 200 Park Avenue New York, NY 10166-0193 Tel: (212) 351-2339 BRosenthal@gibsondunn.com
	Katherine Q. Dominguez (<i>pro hac vice</i> motion requested) Gibson, Dunn & Crutcher LLP 200 Park Avenue New York, NY 10166-0193 Tel: (212) 351-2338 KDominguez@gibsondunn.com
	Timothy P. Best (<i>pro hac vice</i> motion requested) Gibson, Dunn & Crutcher LLP 333 South Grand Avenue Los Angeles, California 90071-1504 Tel: (213) 229-7659 TBest@gibsondunn.com

Petitioners respectfully request authorization to file motions for Josh Krevitt, Brian Rosenthal, Katherine Dominguez, and Timothy Best to appear before the USPTO *pro hac vice*. Mr. Krevitt, Mr. Rosenthal, Ms. Dominguez, and Mr. Best are experienced litigating attorneys and are currently serving as lead counsel for Petitioners in the District Court Action. Each has established familiarity with the

subject matter at issue in this proceeding. Petitioners intend to file a motion to appear *pro hac vice* under 37 C.F.R. § 42.10. Pursuant to 37 C.F.R. § 42.10(b), powers of attorney accompany this petition.

D. Service of Information Under 37 C.F.R. § 42.8(b)(4)

Please address all correspondence to counsel at the addresses above. Petitioners consent to service by email.

III. PAYMENT OF FEES

Per 37 C.F.R. §§ 42.203 and 42.15(b), the required fee was submitted but the Office is authorized to charge any fee deficiency, or credit overpayment, to Deposit Account 50-1408, including additional fees due in connection with this Petition.

IV. REQUIREMENTS OF POST GRANT REVIEW

Per 37 C.F.R. § 42.204, each requirement for PGR of the '869 Patent is satisfied.

A. Standing

Per 37 C.F.R. § 42.204(a), Petitioners certify that the '869 Patent is available for PGR and Petitioners are not barred or estopped from requesting PGR of the claims on the grounds identified herein.

B. Identification of Challenge and Relief Requested

Per 37 C.F.R. § 42.204(b), Petitioners request that the Board institute PGR of claims 1-59 of the '869 Patent on one or more grounds under AIA 35 U.S.C. §§ 102, 103, and 112.

The application leading to the '869 Patent was filed February 14, 2020. It is a continuation of an application filed May 12, 2015. For purposes of this Petition only, Petitioners do not dispute that the Challenged Claims are entitled to an effective filing date of May 12, 2015 (though Petitioners may challenge that in District Court).

Petitioners request that the Challenged Claims be canceled based on the grounds:

Ground	Claims	Basis for Rejection ¹
1	1-59	Obviousness over Ideses
2	1-6, 8-24, 26-27, 29-30, 32-38, 40-41, 43-44, 46-52, 54-55, and 57-58	Obviousness over Mori
3	1-59	Obviousness over Ideses in view of Mori
4	1-59	Indefinite under 112(b) (“the static data points”)
5	8, 14, 17, 29, 43, 57	Indefinite under 112(b) (“character strings” ... “within a predetermined range”)
6	8, 14, 17, 29, 43, 57	Lack of written description under 112(a) (“character strings” ... “within a predetermined range”)

Grounds 1-3 are made in view of the following prior art references, none of which was cited or considered during the '869 Patent's prosecution:

¹ All obviousness grounds include the knowledge of POSITA.

1. Ideses (Ex. 1005)

Ideses, titled “Methods and Systems for Generating Classifiers for Software Applications,” was filed January 29, 2015, and published July 30, 2015. Ideses’s earliest claim to priority on the face of the publication is to a provisional application filed January 30, 2014. Ideses qualifies as prior art under AIA 35 U.S.C. § 102(a)(2). Ideses was neither cited nor considered during the ’869 Patent’s prosecution.

2. Mori (Ex. 1006)

Mori, titled “Software Detection Methods and Devices and Programs,” was filed July 23, 2010, and published February 9, 2012. Mori qualifies as prior art under AIA 35 U.S.C. § 102(a)(1). Mori was neither cited nor considered during the ’869 Patent’s prosecution.

**C. Construction of the Challenged Claims Under 37 C.F.R.
§ 42.204(b)(3)**

A claim subject to PGR shall be construed using the same standard as in a civil action. 37 C.F.R. § 42.200(b); 83 F.R. 51358 (2018). In the Litigation, the Court construed the following terms of the ’844 Patent (’869 Patent’s parent):

Term	Construction
“an executable file”	Plain-and-ordinary meaning
The terms relating to whether “character strings” are “within a predetermined range”	Not indefinite. Plain-and-ordinary meaning
“within a predetermined range”	Plain-and-ordinary meaning, wherein the plain-and-ordinary meaning of “range” is numerical range or a subset of a “particular scale” with at least a

	lower limit or an upper limit, but not necessarily both.
“support vector processing”	Plain-and-ordinary meaning wherein the plain-and-ordinary meaning is “processing that involves a support vector.”
“at least one processor . . .”	Not subject to § 112(f). Plain-and-ordinary meaning.

Ex. 1007, 17-20. This Petition uses the preceding district court constructions where applicable. Patent Owner and Sophos Ltd. have also submitted proposed constructions for the Challenged Claims. Specifically, Sophos Ltd. contends (as explained in Ground 4) that the term “the static data points” is indefinite because it is not clear whether it refers to the “plurality of static data points” or the “identified data points.” Ex. 1008, 52-54. Patent Owner contends that the term refers to the “plurality of static data points.” Ex. 1009, 58-59. In Grounds 1-3, Petitioners explain how the prior art discloses “the static data points” regardless of whether it refers to “plurality of static data points” or “identified data points.”

Petitioners also maintain that the terms relating to whether “character strings” are “within a predetermined range” are indefinite and lack written description as explained in Grounds 5 and 6 below. In the ’844 Patent claim construction proceedings, the District Court did not find the term indefinite but has not yet issued a full claim construction order. Ex. 1007, 2, 18. Because of the District Court’s claim construction for the ’844 Patent, Petitioners did not re-brief this term for the ’869 Patent. Instead, the parties stipulated, while reserving appellate rights, that the

District Court’s claim construction for the ’844 Patent would apply to the ’869 Patent. Ex. 1010. Patent Owner contends that this term is not indefinite and should be given its plain and ordinary meaning. Ex. 1007, 18. In Grounds 1-3, Petitioners explain how the prior art discloses this term under Patent Owner’s and the District Court’s claim construction.

Patent Owner contends that “within a predetermined range” should be given its plain and ordinary meaning without any further elaboration as the District Court provided. Ex. 1009 at 59-60. In Grounds 1-3, Petitioners explain how the prior art discloses this term under Patent Owner’s position and the District Court’s claim construction.

Petitioners do not propose constructions for any other terms because claims only need to be construed “to the extent necessary to resolve the controversy.” *Nidec Motor Corp. v. Zhongshan Broad Ocean Motor Co.*, 868 F. 3d 1013, 1017 (Fed. Cir. 2017). Here, no other terms need construction because the Challenged Claims would have been obvious under any reasonable construction.

D. How the Challenged Claims Are Unpatentable Under 37 C.F.R. § 42.204(b)(4)

An explanation of how the Challenged Claims are unpatentable is provided below.

E. Supporting Evidence Under 37 C.F.R. § 42.204(b)(5)

The supporting evidence is provided below, and a list of exhibits identifying the exhibits is also included in this petition. The technical information and grounds for rejection explained in the petition are further supported by the Expert Declaration of Dr. Gene Tsudik (Ex. 1004). A List of Exhibits is included in this petition per 37 C.F.R. § 42.63(e).

V. THRESHOLD REQUIREMENT FOR PGR

Under 35 U.S.C. § 324(a), institution of PGR requires “the information presented in the petition filed under section 321, if such information is not rebutted, would demonstrate that it is more likely than not that at least one of the claims challenged in the petition is unpatentable.” This petition meets this threshold.

VI. PERSON OF ORDINARY SKILL IN THE ART

A POSITA on May 12, 2015, would have a Bachelor’s degree in Computer Science, Computer Engineering, Electrical Engineering, or an equivalent discipline, and at least one year’s worth of experience relating to static data analysis and malware detection; alternatively, a POSITA could have two or more years of work experience relating to static data analysis and malware detection in lieu of the education requirements. Ex. 1004 ¶ 16.

VII. STATE OF THE ART

All concepts in this section were well-known and widely used by POSITAs at least as of May 12, 2015.

1. Malware

Malware, or malicious software, refers to software developed to modify, steal, or destroy data on a computer or a computer network or to surreptitiously use computer and/or network resources, *e.g.*, crypto-mining malware. Ex. 1004 ¶ 67. These malicious programs operate in a variety of ways. *Id.*, ¶ 68.

Generally, malware is introduced into a computer or computer network through the Internet or an external storage device; it may be hidden within bundles of software, concealed in a web page, or disguised as a recognizable program. *Id.*, ¶ 44; Ex. 1011, 4-6;

2. Static and Dynamic Malware Analysis

Static analysis is the (typically automated) analysis of a target computer program that is performed without executing it—this is in contrast to dynamic analysis, which is performed upon a target program during its (usually sandboxed) execution. Ex. 1004 ¶ 49; Ex. 1012, 6-7; Ex. 1011, 7-8.

VIII. OVERVIEW OF THE '869 PATENT

The '869 Patent purports to describe a method for detecting “whether an executable file is harmful or benign before the executable file is actually executed

on a processing device.” Ex. 1001, 2:12-14. The ’869 Patent and Challenged Claims are directed to using machine-learning models to detect malware. *Id.*, 7:43-8:7, 9:27-54. The ’869 Patent teaches that a “plurality of static data points are extracted from an executable file.” *Id.*, 1:32-33. The ’869 Patent describes generating a “feature vector” from these extracted features. *Id.*, 8:21-34. The ’869 Patent further explains that the number of features collected and analyzed for an executable file can vary, and that the alleged invention “can adaptively set features” of the feature vector “based on the static data points extracted for a file.” *Id.*, 7:43-45; *see id.*, 6:45-7:25. For example, “one or more features of the feature vector may be selectively turned on or off based on evaluation of whether a value of a static data point is within a predetermined range.” *Id.*, 12:67-13:3. Once the feature vector is generated, the Challenged Claims require using machine learning models which may perform “data analysis and pattern recognition” on the feature vector to determine whether the file is malicious. *Id.*, 9:27-54.

A. Summary of the Prosecution History of the ’869 Patent

U.S. application no. 16/791,649 (“the ’649 application”), which issued as the ’869 Patent, was filed February 14, 2020. Ex. 1001, Cover. It is a continuation of application no. 14/709,875 (issued as the ’844 Patent), filed May 12, 2015. As filed, the ’649 application contained one claim. Ex. 1002, 23.

The '869 Patent's parent, the '844 Patent, required several amendments to the claims before issuance, including for example limiting the “extracting a plurality of static data points” step in the independent claims to “wherein the plurality of static data points represent predefined character strings in the executable file.” Ex. 1003, 304-09. The '844 Patent also contained other limitations in the independent claims such as requiring the extraction of the data points without decrypting or unpacking the executable file and limiting the “evaluating the feature vector” step to “using support vector machines.” *Id.* The independent claims of the '869 Patent, meanwhile, do not contain many of these limitations—some, for example, do not limit the static data points to predefined character strings nor limit evaluation to using support vector machines.

After the applicant filed the '649 application, on April 9, 2020, the applicant canceled the sole pending claim and added 20 new claims, of which claims 2, 12, and 18 were independent. Ex. 1002, 63-66.

On March 29, 2021, the examiner issued a nonfinal rejection, rejecting pending claims 2-21 for not complying with the written description and enablement requirements. *Id.*, 71-74. The examiner also rejected pending claims 2-5, 9, 11-16 and 18-20 as being anticipated by “Detection of malicious code by applying machine learning classifiers on static features: A state-of-the-art survey” by Shabtai et al. *Id.*, 74-76. The examiner rejected pending claims 6-8 as obvious over Shabtai and U.S.

Patent Pub. No. 2010/0293273 to Basarrate et al. *Id.*, 76-77. The examiner also rejected claims 2-9 and 9-21 on the grounds of nonstatutory double patenting. *Id.*, 77-79. The examiner noted that dependent claims 10, 17, and 21 would be allowable if rewritten in independent form. *Id.*, 79-80.

In response, the applicant amended claims 2, 10, 12, 17, 18 and 20. *Id.*, 128-31 by adding “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.” *Id.* On August 18, 2021, the examiner issued a notice of allowance. *Id.*, 155-57.

On November 18, 2021, the applicant filed a request for continued examination, amended claims 2, 7, and 8, and added claims 22-51. *Id.*, 171, 174-82. On December 21, 2021, the examiner objected to claim 49 for being duplicative of claim 10, and rejected pending claims 2-5, 9, 11-16, 18-20, 22-28, 33, 35, 37-43, 48, and 50 as being unpatentable over Shabtai and U.S. Patent Pub. No. 2010/0082642 to Forman et al. *Id.*, 194-97. The examiner rejected pending claims 6-8, 30-32, 36, 45-47, and 51 as being unpatentable over Shabtai, Forman, and Basarrate. *Id.*, 197-99. The examiner noted that dependent claims 10, 17, 21, 34, and 49 would be allowable if rewritten in independent form. *Id.*, 199.

In response, the applicant amended claims 2, 12, 22 and 20, and canceled claims 10, 17, 21, 34, and 49. *Id.*, 225. Each of the independent claims were

amended to include “based on at least in part on evaluation of whether a value of one of the plurality of static data points is within a predetermined range.” *Id.*, 217-24. On June 15, 2022, the examiner allowed the claims, and the ’869 Patent issued August 9, 2022. *Id.*, 308, 340.

IX. DETAILED CHALLENGE

A. Ground 1: Claims 1-59 Are Rendered Obvious By Ideses

As described below, Ideses teaches and/or renders obvious each and every Challenged Claims.

1. Prior Art Overview

Ideses discloses “a method for training a classifier for classifying applications.” Ex. 1005, Abstract. Specifically, Ideses discloses a method for providing “run-time classification of software applications, . . . for example, to detect if the software application for installation [on a mobile device] is malware, adware, or benign (e.g., fine for installation).” *Id.*, [0123]. Although Ideses generally describes *run-time* feature extraction (*i.e.*, during execution of a software application), run-time extraction is noted as an *optional* feature, and Ideses discloses extracted features that “may contain static meta data regarding the application, for example, icon, name, rating or other features,” as well as “information regarding the executable code and/or software package, for example, in the form of byte code, resources, permissions, or other features.” *Id.*, [0089]; *see id.*, [0133]. Ideses

discloses that “the extracted features are collated into feature vectors,” and that the subject “software application is classified . . . by a run-time classification module,” which can label the software application, for example, “as benign or adware.” *Id.*, [0134]-[0135].

2. Analysis

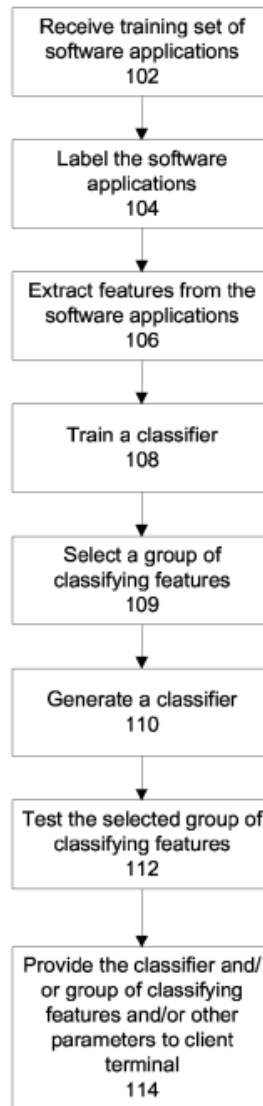
a. Independent Claim 1

[1.pre]: A computer-implemented method comprising:

To the extent the preamble of claim 1 is limiting, Ideses discloses [1.pre]. Ex. 1004 ¶¶ 85-87. Ideses discloses “methods and systems for generating classifiers for software applications.” Ex. 1005, [0002]. Ideses further discloses “computer readable program instructions . . . for causing a processor to carry out aspects of the present invention.” *Id.*, [0063]; *see also id.*, [0064]-[0069].

[1.a]: identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;

Ideses discloses [1.a]. Ex. 1004 ¶¶ 88-100. Figure 1 of Ideses depicts “a method of generating a selected group of classifying features (e.g., feature vector) for classification of software applications”:



Ex. 1005, [0071], Fig. 1.

At step 102, “multiple software applications for training the classifier are received,” which 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, . . . and/or provided by other methods.” *Id.*, [0080]. At step 104, “the multiple software applications are labeled with a classification type based on desired software application classification categories,” which “denote desired or undesired software applications.” *Id.*, [0081]; *see id.*, [0141].

At step 106, “features,” *i.e.*, “static data point that may be indicative of either a harmful or benign executable file” as required by [1.a], “are extracted,” *i.e.*, “identif[ied]” as required by [1.a], “from the software applications, for example, by a feature extractor module 306.” *Id.*, [0081].

The ’869 Patent broadly describes “static data point[s]” as “point[s] of reference used to evaluate an executable file.” Ex. 1001, 5:61-62. The ’869 Patent explains that “static data points include, but are not limited to: header information, section information, import and export information, certificate information, resource information, string and flag information, legal information, comments, and/or program information (e.g., APIs and DLLs), among other examples.” *Id.*, 5:62-67.

Ideses discloses “static data points.” Ex. 1004 ¶¶ 91-93. Specifically, Ideses explains that “the extracted features may contain *static meta data* regarding the application, for example, icon, name, rating or other features,” and may further 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]. The “static meta data” described in Ideses corresponds to the “static data points” required by [1.a], and the examples listed in Ideses match examples of “static data points” identified in the ’869 Patent. For instance:

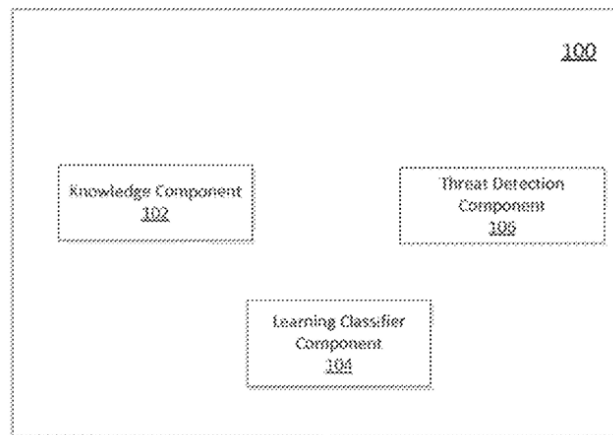
Ideses	’869 Patent
“icon”	“resource information”
“name”	“string . . . information,” “company name,” “internal name,” “product name”
“information regarding the executable code and/or software package . . . [e.g.,] byte code, resources”	“resource information,” “program information (e.g., APIs and DLLs),” “package code”
“permissions”	“file flags”

See id.; Ex. 1001, 5:62-67, Table 1.1; Ex. 1004 ¶ 92.

Ideses further discloses that its features are “indicative of either a harmful or benign executable file” as required by [1.a]. Ex. 1004 ¶¶ 94-95. As Ideses explains, its features “may be any feature that describes the software application,” Ex. 1005, [0089], and the software applications represent “desired or undesired software applications.” *Id.*, [0081]; *see id.*, [0141]. As such features obtained from these applications may reflect whether the application is desired (*e.g.*, “benign”) or undesired (*e.g.*, “harmful”).

The '869 Patent does not use the phrase “knowledge module,” but broadly describes a “knowledge component.” Figure 1 of the '869 Patent depicts an exemplary system:

FIG. 1



Ex. 1001, 1:59-61. The '869 Patent explains that a:

Knowledge component 102 described herein may collect data for use in building, training and/or re-training a learning classifier to evaluate executable files. The knowledge component 102 is ***one or more*** storages, memories, and/or ***modules*** that continuously collects and manages data that may be used to detect threats in files such as executable files.

Id., 3:54-60. The '869 Patent also explains that:

Components of the systems may be hardware components or software implemented on and/or executed by hardware components of the systems . . . For instance, threat detection operations (e.g., application, instructions, modules, etc.) may be

run on a processing device such as a computer, a client device (e.g., mobile processing device, laptop, smartphone/phone, tablet, etc.) and/or any other electronic devices, where the components of the system may be executed on the processing device. In other examples, the ***components*** of systems disclosed herein ***may be spread across multiple devices***.

Id., 3:4-24.

Ideses discloses the claimed “knowledge module” as required by [1.a]. Ex. 1004 ¶¶ 96-100. Figure 3 (depicted below) of Ideses shows “a system 300 for generating a selected group of classifying features for classification, and for classifying based on the selected group. Optionally, system 300 classifies software applications (e.g., adware) on a mobile device 308 during runtime.” Ex. 1005, [0072].

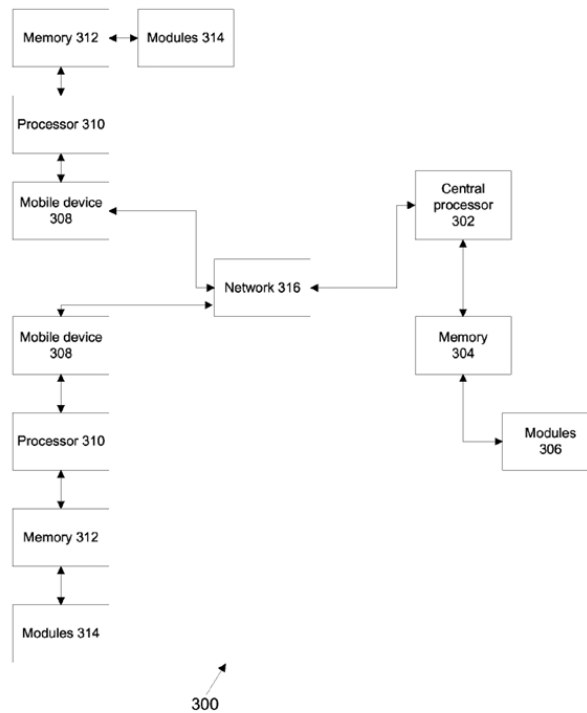


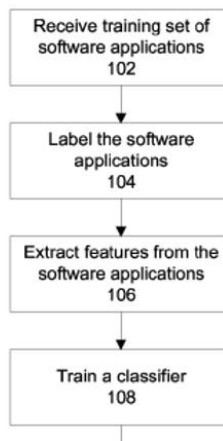
FIG. 3

“[S]ystem 300” includes a central processor, memory, and “modules 306” and “modules 314.” Ex. 1005, [0073]. Ideses discloses that module 306, *i.e.*, “the knowledge module,” can consist of “a labeling module,” “feature extractor module,” and/or “learning module.” *Id.*, [0074], [0081], [0086]. Modules 306 (including the

label, feature extractor, and learning modules) and 314 of Ideses are the claimed “knowledge module.” For example, Ideses discloses that module 306 “may include a feature extraction and/or pruning system.” *Id.*, [0074]; *see also id.*, [0086] (“Optionally, at 106, features are extracted from the software applications, for example, by a feature extractor module 306 stored on memory 304.”). Ideses further discloses that module 314 is a “classification module” and “the classification module labels the software application . . . the software application is classified, for example, as benign or adware . . . the classification module is implemented at the mobile device, or at other client terminals that may or may not be mobile.” *Id.*, [0135]-[0136].

[1.b]: associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

Ideses discloses [1.b]. Ex. 1004 ¶¶ 101-08. As explained in [1.a], Figure 1 (excerpt shown below) of Ideses depicts “a method of generating a selected group of classifying features (e.g., feature vector) for classification of software applications.”



Ex. 1005, [0071], Fig. 1 (excerpt).

Figure 1 shows a classifier being trained based on extracted features and software application labels. *Id.*, Fig. 1. The classifier is subsequently used “for evaluating software applications.” Ex. 1005, [0007]. Ideses also discloses “a labeling module for labeling of software application for generating the classifier.” *Id.*, [0033]. Ideses explains that the labeling module can have “the classification types denote desired or undesired software applications.” *Id.*, [0081]. Ideses explains that the classifier is trained at step 108, “based on the set of extracted features (block 106) and software classification labeling (block 104).” *Id.*, [0094]. A POSITA would have understood, or it would have been obvious from Ideses that training a classifier based on the set of extracted features and software classification labels is associating the features with labels (*i.e.*, categories of files) because Ideses discloses that the classifier is subsequently used to evaluate applications based on

this association. Ex. 1004 ¶ 106; *see* Ex. 1005, [0081]-[0083], [0135]. To be able to evaluate applications, a POSITA would understand that the classifier has previously associated extracted features with software classification labels. Ex. 1004 ¶ 106.

As explained in [1.a], modules 306 and 314 of Ideses are the claimed “knowledge module.” Module 306 of Ideses includes “a learning module 306 (e.g., stored on memory 304) of a machine learning algorithm is applied to train the classifier . . . Optionally, the learning module performs the machine learning and/or classifier training,” *i.e.*, “by the knowledge module” as required by limitation [1.b] *Id.*, [0094]-[0095]. Ideses discloses that “the learning module performs the machine learning and/or classifier training.” *Id.*, [0095]. A POSITA would have understood that module 306 associates “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” as required by [1.b]. Ex. 1004 ¶ 107-08.

[1.c]: identifying an executable file to be evaluated;

Ideses discloses [1.c]. Ex. 1004 ¶¶ 109-14. Figure 2 of Ideses “is a flowchart of a method of run-time execution of a classifier based on a pruned feature list.” Ex. 1005, [0123], Fig. 2. Ideses further discloses that Figure 2 “may provide run-time classification of software applications, the method locally executed by the available

resources on the mobile device, for example, to detect if the software application for installation is malware, adware, or benign (e.g., fine for installation).” *Id.*, [0123].

Ideses discloses in step 206, “a software application,” *i.e.*, an “executable file to be valuated” as required by [1.c], “is received at the mobile device.” *Id.*, [0131]. As explained below in the remaining limitation of claim 1, the software application is evaluated.

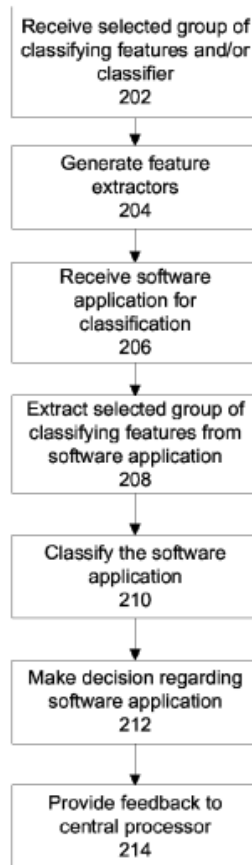
Ideses further discloses that this file is “identif[ied]” as required by [1.c]. The ’869 Patent explains that:

[E]xamples of an identification of an executable file for evaluation may occur in examples such as: when a download of a file is requested, while a download is being performed, evaluation of streaming data, when a new file is detected as attempting to execute (e.g., potentially unwanted file), and before an unknown file or file containing executable code that was not previously checked attempts to execute, among other examples.

Ex. 1001, 7:7-14. Similarly, Ideses explains that a software application is evaluated when, for example, having been “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].

[1.d]: extracting, by the knowledge module, a plurality of static data points from the identified executable file;

Ideses discloses [1.d]. Ex. 1004 ¶¶ 115-23. Figure 2 of Ideses “is a flowchart of a method of run-time execution of a classifier based on a pruned feature list”:



Ex. 1005, [0123], Fig. 2. At step 206, “a software application,” *i.e.*, an “executable file” as required by [1.d], “is received at the mobile device,” for example, having been “downloaded from the internet.” *Id.*, [0131]. At step 208, “features,” *i.e.*, “a plurality of static data points” as required by [1.d], “are extracted based on the software application.” *Id.*, [0132]; Ex. 1004 ¶¶ 116-17.

As explained in [1.a], the ’869 Patent broadly describes “static data points” as “point[s] of reference used to evaluate an executable file.” Ex. 1001, 5:61-62. Ideses

discloses the extraction of such “static data points” from the “software application.”

Ex. 1004 ¶¶ 118-19. As explained in [1.a], the static data points in Ideses match examples of “static data points” identified in the ’869 Patent.

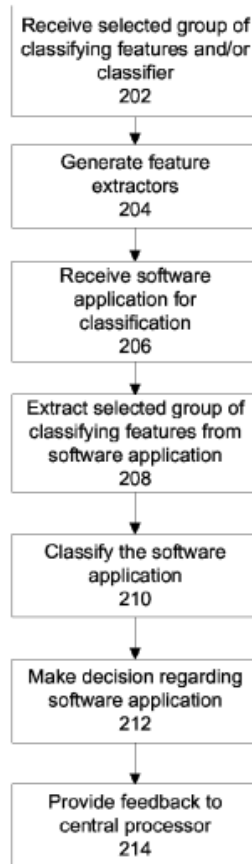
As explained in [1.a], modules 306 and 314 of Ideses are the claimed “knowledge module.” Ideses further discloses that “[t]he method of FIG. 2 may be executed by mobile device 308 of FIG. 3.” Ex. 1005, [0123]. Ideses discloses that “modules 314” is associated with the mobile device and “may include a run-time feature extractor for extracting features from a software application.” *Id.*, [0076]. Alternatively, it would have been obvious to have module 306 perform run-time analysis. Indeed, in Ideses, Figure 3 is an example of a mobile device, but in other examples, such as a computer connected to a network, modules 306 could perform both classifier training and run-time analysis. Ex. 1004 ¶¶ 120-23; *see* Ex. 1005, [0135]-[0136].

[1.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

This limitation includes multiple parts, each of which is disclosed by Ideses. Ex. 1004 ¶¶ 124-36.

Generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points: Ideses discloses this portion of

[1.e]. Ex. 1004 ¶¶ 74-75. As explained above with respect to [1.d], Figure 2 of Ideses “is a flowchart of a method of run-time execution of a classifier based on a pruned feature list”:



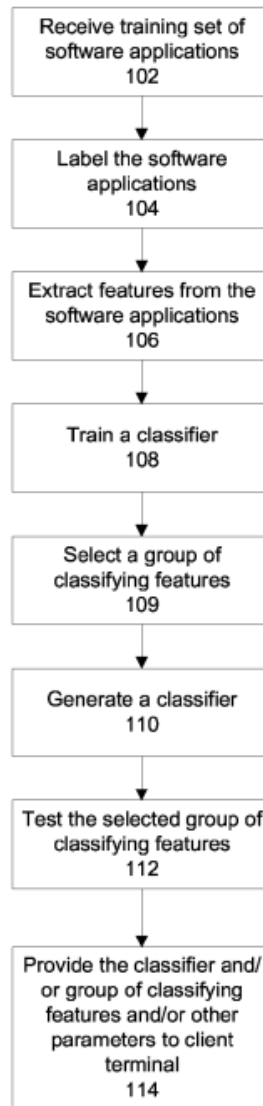
Ex. 1005, [0123], Fig. 2. At step 202, “*the trained classifier . . .* is received by the mobile device.” *Id.*, [0124].

If “the static data points” refers to “the plurality of static data points,” *see* Ground 4, Ideses explains that the classifier used at step 208 of Figure 2 is the

“trained classifier received from the central server” in block 202. Ex. 1005, [0132]. If “the static data points” refers to “the identified static data points,” *see* Ground 4, Ideses discloses, as explained in the next paragraph, that the “trained classifier received from the central server” is used to evaluate the generate the feature vector. Ex. 1005, [0132]. As explained in [1.b], Ideses’ classifier received from the central server is trained using “the identified static data points.” A POSITA would understand that training a classifier using static data points results in “classifier trained to classify [those] static data points” as required by [1.e]. Ex. 1004 ¶ 126.

Ideses further explains that “the extracted features are collated into *feature vectors*.” *Id.*, [0134]. Ideses therefore discloses “generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points,” as required by [1.e]. Ex. 1004 ¶ 127.

Based on training data, the training data comprising files known to fit into one of the plurality of categories of files: Ideses discloses this portion of [1.e]. Ex. 1004 ¶¶ 128-31. Figure 1 of Ideses depicts “a method of generating a selected group of classifying features (e.g., feature vector) for classification of software applications,” *i.e.*, a “classifier” as claimed:



Ex. 1005, [0071], Fig. 1. At step 102, “multiple software applications for training the classifier are received,” which 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, . . . and/or provided by other methods.” *Id.*, [0080].

At step 104, “the multiple software applications are labeled with a classification type based on desired software application classification categories,” which “denote *desired* or *undesired* software applications.” *Id.*, [0081]. Ideses also mentions “Adware, Goodware, Intrusive Adware,” as other classification types that may be used. *Id.*, [0083]; *see id.*, [0141] (“[T]he software application is installed on the mobile device, for example, if the classification type is determined to be benign, or other non-harmful and/or beneficial classification types. Alternatively, the 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.”). These classification types correspond to the “plurality of categories of files” (*e.g.*, “harmful” or “malware,” “undesired,” such as “Adware,” “Intrusive Adware,” “bothersome,” or “spyware,” and “desired,” such as “benign, or other non-harmful and/or beneficial classification types”). *See* Ex. 1004 ¶ 130.

At step 106, “features are extracted from the software applications,” including, for example, “data and/or meta data.” *Id.*, [0086]-[0089]. And, at step 108, “a classifier is trained based on the set of extracted features (block 106) and

software classification labeling (block 104).” *Id.*, [0094]. Ideses therefore discloses “generating a feature vector . . . based on training data, the training data comprising files known to fit into one of the plurality of categories of files,” as required by this portion of [1.e]. Ex. 1004 ¶ 131.

Wherein one or more features of the feature vector are selectively turned on or off 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. Ideses discloses this portion of [1.e]. Ex. 1004 ¶¶ 132-36. Ideses explains that in the method depicted in Figure 1, “[t]he full feature list is optionally selectively pruned,” *i.e.*, “selectively turned off” as required by [1.e], “to generate a pruned feature list.” Ex. 1005, [0071]. Further, “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,” *i.e.*, “selectively turned on” as required by [1.e]. *Id.*

Ideses explains that “[t]he group of classifying features may be selected based on one or more methods,” and that “the group of classifying features is selected to reduce the dimensionality of the feature vector.” *Id.*, [0104]-[0105]. For example, the group of classifying features may be selected “based on the significance of the extracted features,” such that significant features, or “those that have a large effect on the classification outcomes, may be retained,” whereas insignificant features, or

“those that have a small, negligible or no effect on classification outcome may be pruned.” *Id.*, [0106].

Ideses provides two approaches to pruning insignificant features. First, “features that correspond to coefficients having a value of zero are pruned.” *Id.*, [0107]. Second, “features with a low magnitude value of the coefficient are identified for pruning and/or are not included.” *Id.* Ideses explains that the selection of the group of classifying features based on coefficients with zero/low values may be denoted by the following expressions, respectively:

$$\text{Feature}_i \in FVf / (\text{Coefficient}_i) \neq 0$$

$$\text{Feature}_i \in FVf / (\text{Coefficient}_i) \geq \alpha$$

Id., [0109]-[0110]. Both of these approaches to selectively including/excluding features are based on whether a feature coefficient “is within a predetermined range” as required by this portion of [1.e], and consistent with the district court’s construction of “within a predetermined range.” A POSITA would understand that “features with a low magnitude” necessarily means there is an upper limit that sets the range for the “low” magnitude. Ex. 1004 ¶ 135. Moreover, Ideses discloses that such an approach can be used to generate a “pruned feature vector” (*i.e.*, a vector in which features of the feature vector are selectively turned on or off) for classification by a classifier. Ex. 1005, [0157], Fig. 5; Ex. 1004 ¶ 136. Ideses therefore discloses

“generating a feature vector . . . herein one or more features of the feature vector are selectively turned on or off 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,” as required by this portion of [1.e].

[1.f]: providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

Ideses discloses [1.f]. Ex. 1004 ¶¶ 137-42. At step 210 of Ideses’ Figure 2, “the software application is classified,” for example, “as benign or adware.” Ex. 1005, [0135]. Ideses explains that “the software application is classified by applying the trained classifier received from the central server,” and that alternatively or additionally, “the software application is classified based on the group of classifying features that have been computed by the feature extractor” at step 208. *Id.*, [0137].

Ideses explains that the classifier may be “trained based on supervised learning,” an example of which are “Support Vector Machines.” *Id.*, [0096]; *see id.*, [0107] (“[F]eatures 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).”). A POSITA would have understood Ideses’ disclosure of classification “based on a support vector machine” to be synonymous with “providing the generated feature vector to one or more support vector machines” as required by [1.f]. *See* Ex. 1004 ¶ 140.

Moreover, a POSITA would have understood that the “support vector machine” described in Ideses would have been used to build a probabilistic model to classify the files as required by [1.f], because a probabilistic model would be necessary to “compute[] the most likely class” in a “multi-class scenario.” Ex. 1005, [0138]-[0141]; *see* Ex. 1004 ¶ 142. In a base use case, SVMs are used as non-probabilistic binary classifiers, meaning that they are capable of classifying a new example into one of two classes. *Id.* However, a well-known technique called “Platt scaling” can be used to allow an SVM to predict a probability distribution over a set of classes, rather than between two. *Id.*; *see, e.g.*, Ex. 1013 at 9 (“This chapter presents a method for extracting probabilities $P(\text{class}|\text{input})$ from SVM outputs, which is useful for classification post-processing.”). It would have been obvious to a POSITA to use Platt scaling with the SVM technique described in Ideses in order to allow the SVM to classify a software application into additional categories, such as “harmful, malware, bothersome, adware, intrusive, spyware, or other non-desirable classification types” in a “multi-class scenario.” Ex. 1005, [0139], [0141]; *see* Ex. 1004 ¶ 142.

b. Claim 2

[2]: The computer-implemented method according to claim 1, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

Ideses discloses and/or renders obvious [2]. Ex. 1004 ¶¶ 143-46. Regarding the extraction step shown in Ideses' Figure 2, (*see* [1.d] above), Ideses explains that “[o]ptionally, features are extracted during run-time,” Ex. 1005, [0133], and Ideses also discloses conversely that “[t]he feature extraction may take place off-line, not part of a run-time operation.” *Id.*, [0090]. While run-time feature extraction would require decryption or unpacking the executable file, off-line feature extraction would be conducted without decrypting or unpacking the executable file. Ex. 1004 ¶ 143-46.

Ideses explains that “[t]he extracted features may be varied, containing information from different modalities,” and distinguishes between “static meta data” (which can be collected “off-line”) and “behavioral features, for example, temporal information regarding system calls, system usage, CPU usage, [and] network utilization,” which can only be extracted at “run-time,” (during which the executable file is necessarily decrypted and/or unpacked). *See* Ex. 1004 ¶ 142.

Ideses lists a number of features that may be extracted, including “*static meta data* regarding the application, for example, icon, name, rating[,] . . . 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]. It would have been obvious to a POSITA reading Ideses to perform “off-line” extraction of the “static meta data” without attempting to first decrypt or unpack the executable file. Ex. 1004 ¶¶ 143-46.

Indeed, it was well known prior to the effective filing date of the ’869 Patent that a “basic” static malware analysis technique was to search through a file’s strings “to get hints about the functionality of a program,” and that “[m]alware that is packed or obfuscated contains very few strings.” Ex. 1012, 8-10; Ex. 1004 ¶ 146; *see also*, e.g., Ex. 1010, 12 (explaining that an “unpacking stub can be viewed by a malware analyst” and that “[i]f [a malware analyst] attempt[s] to perform static analysis on the packed program, [the analyst] will be analyzing the stub, not the original [malware] program”). Thus, it would have been obvious to a POSITA that extracting “static meta data” from a potential malware executable without first unpacking or decrypting the executable would have been the first step in a basic static analysis. Ex. 1004 ¶ 146.

c. Claim 3

[3] The computer-implemented method according to claim 1, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

Ideses discloses and/or renders obvious [3]. Ex. 1004 ¶¶ 147-50. As explained in [1.f], Ideses discloses “providing the generated feature vector to one or

more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.”

A POSITA would have understood that “support vector machines” that use Platt scaling to build a probabilistic model would have utilized data analysis and pattern recognition to generate a feature vector. *See* Ex. 1004 ¶¶ 149-50. Platt scaling was used to “train the parameters of an additional sigmoid function to map the SVM outputs into probabilities,” and then the SVM+sigmoid combination was used to determine the probabilities that a given example belongs to each category. *See id.*; *see* Ex. 1013, 1, 3-7. It would have thus been obvious to a POSITA that an SVM “builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors” when it is used with Platt scaling by fitting a two-parameter sigmoid function that maps classification probabilities to the SVM outputs (*i.e.*, the “feature vectors”). *See id.*; *see* Ex. 1004 ¶ 149-50.

d. Claim 4

[4] The computer-implemented method according to claim 1, wherein the probabilistic model indicates whether the executable file is harmful.

Ideses discloses [4]. Ex. 1004 ¶¶ 151-53. As explained in [1.f], at step 210 of Ideses’ Figure 2, “the software application is classified,” for example, “as benign or adware.” Ex. 1005, [0135]. A POSITA would have understood that classifying “as benign or adware” indicates “whether the executable file is harmful” as required by claim [4]. Ex. 1004 ¶¶ 151-53.

e. Claim 5

[5] The computer-implemented method according to claim 1, wherein the executable file is identified in response to a detected condition.

Ideses discloses and/or renders obvious [5]. Ex. 1004 ¶¶ 154-57. As explained above with respect to [1.c], Ideses discloses “identifying an executable file to be evaluated.” Further, and as also explained above with respect to [1.c], Ideses discloses non-limiting examples of when a file can be identified, for example, when it has been “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]. A POSITA would understand that each of these examples is a “detected condition,” as required by claim [5]. Ex. 1004 ¶ 157.

f. Claim 6

[6] The computer-implemented method according to claim 5, wherein the detected condition is user request for a file download.

Ideses discloses and/or renders obvious [6]. Ex. 1004 ¶¶ 158-60. As explained above with respect to [1.c] and [5], Ideses discloses that a file can be identified when it has been “downloaded from the Internet” or “other methods of receipt.” A POSITA would understand, or it would be obvious, that a file could be identified when a user requests for a file to be downloaded in addition to after a file has been downloaded. Ex. 1004 ¶¶ 158-60.

g. Claim 7

[7] The computer-implemented method according to claim 5, wherein the detected condition is the detection of a new file attempting to execute.

Ideses discloses and/or renders obvious [7]. Ex. 1004 ¶¶ 161-63. As explained above with respect to [1.c] and [5], Ideses discloses that a file can be identified when it has been “loaded using physical computer readable media” or when a “software application is requesting (automatically or manually) to be installed.” A POSITA would understand, or it would be obvious, that a file could be identified when it attempts to install (*i.e.*, execute), or when it attempts to run after installation. Ex. 1004 ¶¶ 161-63.

h. Claim 8

[8] The computer-implemented method according to claim 1, wherein the plurality of static data points represent predefined character strings in the executable file.

Ideses discloses and/or renders obvious [8]. Ex. 1004 ¶¶ 164-66. Ideses explains that “[t]he extracted features *may be any feature that describes the software application*,” including “static meta data regarding the application, for example, *icon, name*, rating or other features,” and 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]. A POSITA would have understood, or it would have been obvious from Ideses that “any feature that describes the software application” would include “predefined

character strings” in the executable file. *See* Ex. 1004 ¶ 166. Several of the example features listed in Ideses, such as the application “name” and “information regarding . . . the software package” would comprise “predefined character strings.” *Id.*

i. Claim 9

[9] The computer-implemented method according to claim 1, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

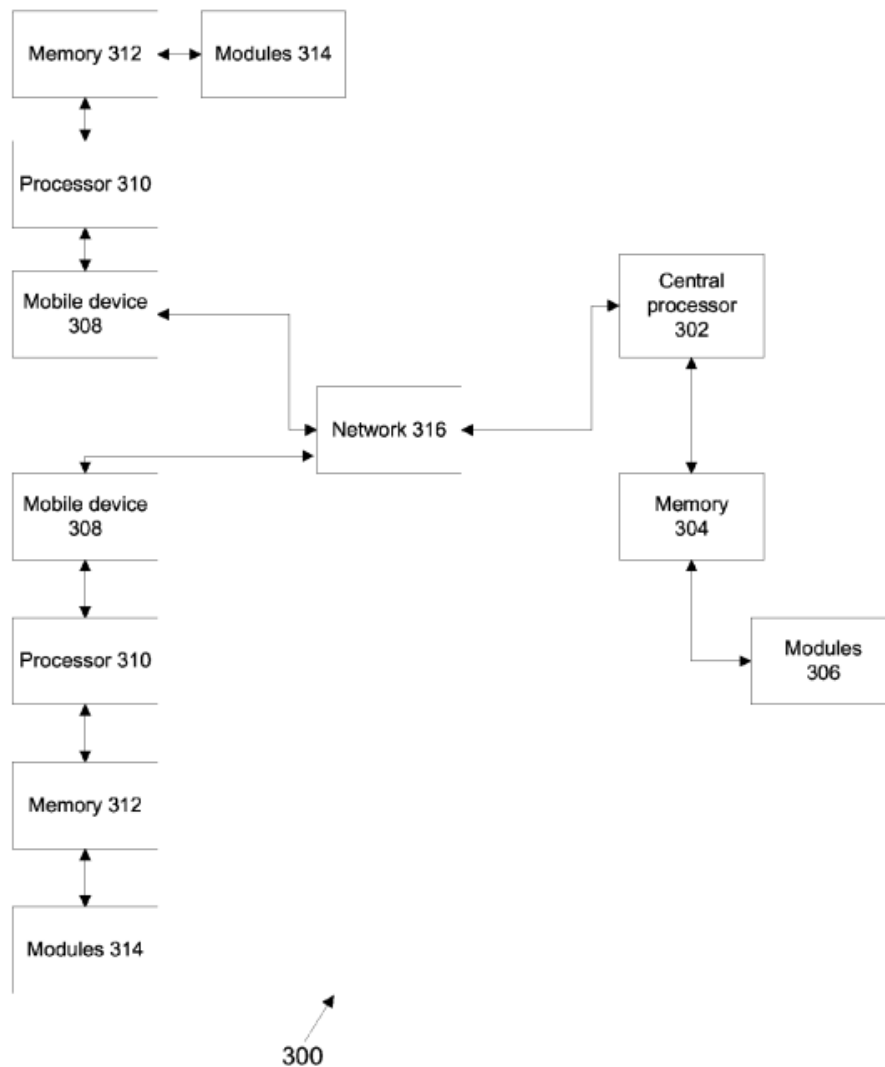
Ideses discloses and/or renders obvious [9]. Ex. 1004 ¶¶ 167-69. Ideses explains that following classification (step 210 of Figure 2), at step 214, “the mobile device reports back to the server,” and “the classification type outcome” (*i.e.*, “harmful, malware, bothersome, adware, intrusive, spyware,” etc.) is reported. Ex. 1005, [0144]. Ideses explains that “the central server learns about the existence of new software applications based on the feedback provided from the mobile device,” and that “the central server re-labels or confirms the labeling of existing software applications based on the provided feedback.” *Id.*, [0145]-[0146]. As an example, “classification results with high certainty received from multiple mobile devices may cause the central server to change the existing classification type, or to retain the existing classification type.” *Id.*, [0146].

j. Independent Claim 10

[10.pre]: A system comprising: at least one memory; and at least one processor operatively connected with the memory and configured to perform operation of:

To the extent the preamble of claim 10 is limiting, Ideses discloses [10.pre].

Ex. 1004 ¶¶ 171-73. Ideses discloses “methods and systems for generating classifiers for software applications.” Ex. 1005, Cover. Specifically, Ideses discloses “a system 300 for generating a selected group of classifying features for classification, and for classifying based on the selected group.” Ex. 1005, [0072].



Id., Fig. 3. Ideses explains that “[s]ystem 300 includes a central processor 302 with a memory 304 for storing data modules 306 thereon.” *Id.*, [0073]; *see id.*, [0075] (“Mobile devices 308 may be resource limited, having smaller and/or less powerful

processor 310, and/or smaller and/or slower memory 312. Modules 314 are stored on memory 312 for execution by processor 310.”).

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

See claim [1.a].

[10.b]: 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;

See claim [1.b].

[10.c]: identifying an executable file to be evaluated;

See claim [1.c].

[10.d]: extracting a plurality of static data points from the identified executable file; and

See claim [1.d].

[10.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

See claim [1.e].

[10.f]: providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

See claim [1.f].

k. Claim 11

[11]: The system according to claim 10, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

See claim [2].

l. Claim 12

[12]: The system according to claim 10, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.

See claim [3].

m. Claim 13

[13]: The system according to claim 10, wherein the probabilistic model indicates whether the executable file is harmful.

See claim [4].

n. Claim 14

[14]: The system according to claim 10, wherein the plurality of static data points represent predefined character strings in the executable file.

See claim [8].

o. Independent Claim 15

[15.pre]: A computer-readable storage device containing instructions, that when executed on at least one processor, causing the processor to execute a process comprising:

See claim [1.pre].

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

See claim [1.a].

[15.b]: 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;

See claim [1.b].

[15.c]: identifying an executable file to be evaluated;

See claim [1.c].

[15.d]: extracting a plurality of static data points from the identified executable file;

See claim [1.d].

[15.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

See claim [1.e].

[15.f]: providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

See claim [1.f].

p. Claim 16

[16]: The computer-readable storage device according to claim 15, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

See claim [2].

q. Claim 17

[17]: The computer-readable storage device according to claim 15, wherein the plurality of static data points represent predefined character strings in the executable file.

See claim [8].

r. Independent Claim 18

[18.pre]: A computer-implemented method comprising:

See claim [1.pre].

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

See claim [1.a].

[18.b]: 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;

See claim [1.b].

[18.c]: identifying an executable file to be evaluated;

See claim [1.c].

[18.d]: extracting a plurality of static data points from the executable file;

See claim [1.d].

[18.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

See claim [1.e].

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

See claim [1.f].

s. Claim 19

[19]: The computer-implemented method according to claim 18, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

See claim [2].

t. Claim 20

[20]: The computer-implemented method according to claim 18, wherein the machine learning model comprises an artificial neural network.

Ideses discloses [20]. Ex. 1004 ¶¶ 202-04. As explained above in [1.f], at step 210 of Ideses' Figure 2, "the software application is classified." Ideses explains that the classifier may be "trained," an example of which are "Neural Networks." Ex. 1005, [0096].

u. Claim 21

[21]: The computer-implemented method according to claim 18, wherein the machine learning model comprises a support vector machine.

See claim [1.f].

v. Claim 22

[22]: The computer-implemented method according to claim 18, wherein the machine learning model comprises a machine learning decision tree.

Ideses discloses [22]. Ex. 1004 ¶¶ 206-08. As explained above in [1.f], at step 210 of Ideses' Figure 2, "the software application is classified." Ideses explains

that the classifier may be “trained,” an example of which are “Decision Trees.” Ex. 1005, [0096].

w. Claim 23

[23]: The computer-implemented method according to claim 18, wherein the machine learning model comprises a Bayesian network.

Ideses discloses [23]. Ex. 1004 ¶¶ 209-11. As explained above in [1.f], at step 210 of Ideses’ Figure 2, “the software application is classified.” Ideses explains that the classifier may be “trained,” examples of which are “Neural Networks, Support Vector Machines, Decision Trees, Hard/Soft Thresholding, Naive Bayes Classifiers, or any other suitable classification system and/or method.” Ex. 1005, [0096]. A POSITA would have understood Ideses’ disclosure of “Naive Bayes Classifiers” and “other suitable classification system/and or method” to include a “Bayesian network.” See Ex. 1004 ¶ 211.

x. Claim 24

[24]: The computer-implemented method according to claim 18, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

See claim [1.f].

y. Claim 25

[25]: The computer-implemented method according to claim 18, wherein the machine learning model comprises a clustering model.

Ideses discloses [25]. Ex. 1004 ¶¶ 213-15. As explained above in [1.f], at step 210 of Ideses’ Figure 2, “the software application is classified.” Ideses explains

that the classifier may be “trained,” an example of which are “k-Nearest Neighbors (KNN) clustering.” Ex. 1005, [0096]. A POSITA would have understood Ideses’ disclosure of KNN clustering to include a “clustering model.” See Ex. 1004 ¶ 215.

z. Claim 26

[26]: The computer-implemented method according to claim 18, wherein the executable file is identified in response to a detected condition.

See claim [5].

aa. Claim 27

[27]: The computer-implemented method according to claim 26, wherein the detected condition is user request for a file download.

See claim [6].

bb. Claim 28

[28]: The computer-implemented method according to claim 26, wherein the detected condition is the detection of a new file attempting to execute.

See claim [7].

cc. Claim 29

[29]: The computer-implemented method according to claim 18, wherein the plurality of static data points represent predefined character strings in the executable file.

See claim [8].

dd. Claim 30

[30]: The computer-implemented method according to claim 18, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

See claim [9].

ee. Claim 31

[31]: The computer-implemented method according to claim 18, further comprising preventing execution of the executable file based on classifying the executable file as harmful.

Ideses discloses this additional element. Ex. 1004 ¶¶ 221-24. Ideses discloses that at step 210 of Figure 2, “a certainty of the classification [of the software application] is provided, for example, the estimated probability that the classification is correct.” Ex. 1005, [0140]. Ideses explains that the “decision to install or remove the software application may be based on the certainty level,” and that “high probability malicious software may be automatically (or manually) removed.” *Id.*, [0143]. Removal of the “high probability malicious software” corresponds to “preventing execution of the executable file” as required by claim [31]. Ex. 1004 ¶ 224.

ff. Independent Claim 32

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

See claim [1.pre].

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

See claim [1.a].

[32.b]: 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;

See claim [1.b].

[32.c]: identifying an executable file to be evaluated;

See claim [1.c].

[32.d]: extracting a plurality of static data points from the executable file;

See claim [1.d].

[32.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

See claim [1.e].

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

See claim [1.f].

gg. Claim 33

[33]: The computer-readable storage device according to claim 32, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

See claim [2].

hh. Claim 34

[34]: The computer-readable storage device according to claim 32, wherein the machine learning model comprises an artificial neural network.

See claim [20].

ii. Claim 35

[35]: The computer-readable storage device according to claim 32, wherein the machine learning model comprises a support vector machine.

See claim [1.f].

jj. Claim 36

[36]: The computer-readable storage device according to claim 32, wherein the machine learning model comprises a machine learning decision tree.

See claim [22].

kk. Claim 37

[37]: The computer-readable storage device according to claim 32, wherein the machine learning model comprises a Bayesian network.

See claim [23].

ll. Claim 38

[38]: The computer-readable storage device according to claim 32, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

See claim [1.f].

mm. Claim 39

[39]: The computer-readable storage device according to claim 32, wherein the machine learning model comprises a clustering model.

See claim [25].

nn. Claim 40

[40]: The computer-readable storage device according to claim 32, wherein the executable file is identified in response to a detected condition.

See claim [5].

oo. Claim 41

[41]: The computer-readable storage device according to claim 40, wherein the detected condition is user request for a file download.

See claim [6].

pp. Claim 42

[42]: The computer-readable storage device according to claim 40, wherein the detected condition is the detection of a new file attempting to execute.

See claim [7].

qq. Claim 43

[43]: The computer-readable storage device according to claim 32, wherein the plurality of static data points represent predefined character strings in the executable file.

See claim [8].

rr. Claim 44

[44]: The computer-readable storage device according to claim 32, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

See claim [9].

ss. Claim 45

[45]: The computer-readable storage device according to claim 32, wherein the process further comprises preventing execution of the executable file based on classifying the executable file as harmful.

See claim [31].

tt. Independent Claim 46

[46.pre]: A system comprising: at least one memory; at least one processor configured to perform operation of:

See claim [10.pre].

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

See claim [1.a].

[46.b]: 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

See claim [1.b].

[46.c]: at least one processor configured to perform operation of: identifying an executable file to be evaluated;

See claims [10.pre] and [1.c].

[46.d]: extracting a plurality of static data points from the executable file;

See claim [1.d].

[46.e]: generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

See claim [1.e].

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

See claim [1.f].

uu. Claim 47

[47]: The system according to claim 46, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.

See claim [2].

vv. Claim 48

[48]: The system according to claim 46, wherein the machine learning model comprises an artificial neural network.

See claim [20].

ww. Claim 49

[49]: The system according to claim 46, wherein the machine learning model comprises a support vector machine.

See claim [1.f].

xx. Claim 50

[50]: The system according to claim 46, wherein the machine learning model comprises a machine learning decision tree.

See claim [22].

yy. Claim 51

[51]: The system according to claim 46, wherein the machine learning model comprises a Bayesian network.

See claim [23].

zz. Claim 52

[52]: The system according to claim 46, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.

See claim [1.f].

aaa. Claim 53

[53]: The system according to claim 46, wherein the machine learning model comprises a clustering model.

See claim [25].

bbb. Claim 54

[54]: The system according to claim 46, wherein the executable file is identified in response to a detected condition.

See claim [5].

ccc. Claim 55

[55]: The system according to claim 54, wherein the detected condition is user request for a file download.

See claim [6].

ddd. Claim 56

[56]: The system according to claim 54, wherein the detected condition is the detection of a new file attempting to execute.

See claim [7].

eee. Claim 57

[57]: The system according to claim 46, wherein the plurality of static data points represent predefined character strings in the executable file.

See claim [8].

fff. Claim 58

[58]: The system according to claim 46, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

See claim [9].

ggg. Claim 59

[59]: The system according to claim 46, further comprising at least one processor configured to perform operation of: preventing execution of the executable file based on classifying the executable file as harmful.

See claim [31].

B. Ground 2: Claims 1-6, 8-24, 26-27, 29-30, 32-38, 40-41, 43-44, 46-52, 54-55, and 57-58 Are Rendered Obvious By Mori

As described below, Mori teaches and/or renders obvious each and every element of claim 1-6, 8-24, 26-27, 29-30, 32-38, 40-41, 43-44, 46-52, 54-55, and 57-58.

1. Prior Art Overview

Mori describes “a software detection device that detects malicious software (hereinafter referred to as ‘malware’).” Ex. 1006, [0010]. Mori explains that “[m]alware often uses deliberately complex encryption and compression (called packing) to avoid detection by pattern matching or static analysis by inverse assemblers,” and that, in general, “in order to perform a detailed analysis of malware,

packed executable files need to be decrypted using various techniques.” *Id.*, [0003]-[0004]. Mori notes, though, that methods “that use decryption, disassembly, and debuggers are time-consuming and require computational resources, and so are not suitable for use in detecting communications containing malware in real time.” *Id.*, [0007].

To solve this problem, Mori “focuses on readable strings contained in software files, collects information (characteristic vectors) that characterizes malware, such as messages contained in programs that constitute malware, host names, execution command names, function names and library names used for communication,” and “compares this content with strings contained in files being inspected to detect malware.” *Id.*, [0023]. In particular, Mori discloses a malware detection method that includes a “a word extraction means that extracts as a word a string of printed and readable bytes” from a potential malware object, “a feature extraction step wherein the feature extraction means configures feature vectors for each [potential malware] Object based on a set of words extracted by the word extraction step,” and a determination as to “whether or not the Object is malware based on newly observed feature vectors by applying a machine learning system to the feature vector.” *Id.*, [0016].

2. Analysis

a. Independent Claim 1

[1.pre]

To the extent the preamble of claim 1 is limiting, Mori discloses [1.pre]. Ex. 1004 ¶¶ 248-50. Mori discloses “a software detection method, apparatus and program for detecting malware or communications transmitting and/or receiving malware using supervised machine learning systems.” Ex. 1006, [0001]. And, “[t]he operation of each component of the detection device shown in FIG. 1 above can be programmatically constructed and then installed and executed on a computer utilized as the detection device.” *Id.*, [0067].

[1.a]

Mori discloses [1.a]. Ex. 1004 ¶¶ 251-63.

Static data points that may be indicative of either a harmful or benign executable file: As explained below, Mori describes supervised training methods, in which teacher data is used. The teacher consists of a feature vector, which is constructed based on byte strings (*i.e.*, “static data points” as required by [1.a]), and a classification of either “0” for a file or “1” for malware (*i.e.*, “a harmful or benign executable file,” as required by [1.a]).

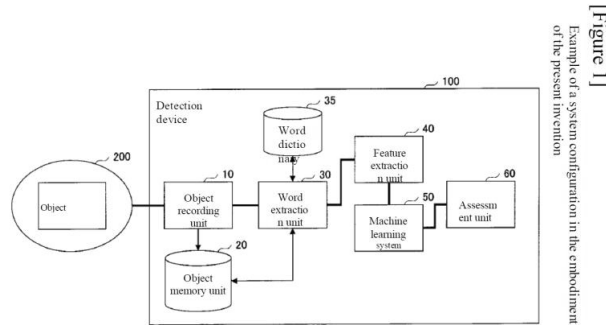
Mori describes “supervised learning methods can be used” for “machine learning.” Ex. 1006, [0056]. Mori explains that “teacher data . . . is given in advance, with V_i serving as a feature vector . . . and y_i is of the same file class and

has a value of ‘0’ or ‘1’” with “0” defined as a file and “1” defined as malware. *Id.* A POSITA would understand that the feature vector used in the teacher data is constructed from static data points. Ex. 1004 ¶¶ 252-59. Indeed, Mori also explains that the feature vector is created from byte strings (*i.e.*, “static data points”). To create a feature vector, Mori explains that a file or communication “is recorded by the Object recording unit.” Ex. 1006, [0028]. Next, “a word extraction unit . . . extracts the printed and readable byte strings as words from the Object recorded above, and uses it as a word set.” *Id.*, [0029]; *see id.*, [0025]. Then, “a feature vector” is constructed “for each Object based on the above word set.” Ex. 1006, [0029]-[0030].

Mori further explains that the feature vectors in the teacher data are “indicative of either a harmful or benign executable file,” as required by [1.a]. Ex. 1004 ¶ 258. Mori explains that each feature vector is associated with a “1” or “0,” where “‘0’ is usually defined as a file, and ‘1’ is defined as malware.” *Id.*, [0056]. A POSITA would understand that the association of the feature vectors to a “1” or “0” can be done using an assessment unit. Ex. 1004 ¶ 259. Indeed, Mori also explains that the “assessment unit” applies machine learning to “the feature vector” and “the file is classified as ‘malware’ or ‘normal file.’” Ex. 1006, [0054]-[0055].

Identifying, by a knowledge module: As explained in Ground 1, [1.a], the ’869 Patent does not use the phrase “knowledge module” but broadly describes a

“knowledge component.” Mori discloses such a “knowledge module” as required by [1.a]. Figure 1 of Mori shows a detection device with a “word extraction unit,” “feature extraction unit,” “machine learning system,” and “assessment unit.”



Ex. 1006, Fig. 1. Mori’s “word extraction unit,” “feature extraction unit,” “machine learning system,” and “assessment unit” constitute the claimed “knowledge module.” As explained in the preceding paragraphs, these units are used to identify the static data points. Ex. 1004 ¶¶ 260-63.

[1.b]

Mori discloses [1.b]. Ex. 1004 ¶¶ 264-68. As explained in [1.a], Mori discloses teacher data which is the association of a feature vector (*i.e.*, “the identified static data points”) with a “1” or “0” (*i.e.*, “one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files”) by the assessment unit (part of the “knowledge module”).

[1.c]

Mori discloses [1.c]. Ex. 1004 ¶¶ 269-74. Mori explains that in the first step of its malware detection method, “the file or communication data to be sent or received (hereinafter, the file or communication data is collectively referred to as the ‘Object’) is recorded by the Object recording unit” Ex. 1006, [0028]. Determining a file is to be sent or received is “identifying an executable file to be evaluated.”

[1.d]

Mori discloses [1.d]. Ex. 1004 ¶¶ 275-77. After an object is recorded by the object recording unit, “a word extraction unit . . . extracts the printed and readable byte strings as words from the Object recorded above, and uses it as a word set.” *Id.*, [0029]; *see id.*, [0025] (“Readable strings include messages contained in programs that make up malware, host names used for communication, execution command names, function names, library names, and so on, and can collect a variety of information about malware.”). As explained in [1.a], the “word extraction unit” is part of the claimed “knowledge module.” Mori explains that its method takes advantage of the fact that “[a]lthough the majority of malware today is known to develop a large number of variants after the original is developed,” that “*it is likely that the characteristics of the printable strings [in a variant] will be similar to the original.*” *Id.*, [0026].

[1.e]

This limitation includes multiple parts, each of which is disclosed by Mori.
Ex. 1004 ¶¶ 278-93.

Generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points. Mori discloses this portion of [1.e]. *Id.* ¶¶ 279-81. If “the static data points” refers to “the plurality of static data points,” *see* Ground 4, Mori explains that after the “word extraction unit . . . extracts the printed and readable byte strings as words from the Object recorded above, and uses it as a word set,” that “*a feature vector*” is constructed “for each Object based on the above word set.” Ex. 1006, [0029]-[0030]. Mori also discloses that “known supervised learning methods can be used” wherein “[t]eacher data . . . is given in advance,” which can then be used as “a highly accurate method for estimating [a] class . . . for the newly observed” feature vector. *Id.*, [0056]-[0057]; *see id.*, [0031] (“The supervised machine learning system outlined above can be re-learned by providing feedback of new data each time an arbitrary configurable cycle arrives.”).

If “the static data points” refers to “the identified static data points,” *see* Ground 4, Mori discloses, as explained in the next paragraph, that the “supervised machine learning system” is used to evaluate the generate the feature vector. As explained in [1.a], Mori’s machine learning system is trained using “the identified static data points.” A POSITA would understand that training a machine learning

system using static data points results in a “classifier trained to classify [those] static data points.” Ex. 1004 ¶ 281.

Based on training data, the training data comprising files known to fit into one of the plurality of categories of files. Mori discloses this portion of [1.e]. *Id.*, ¶¶ 282-84. As explained above, Mori discloses that “known supervised learning methods” can be used wherein “teacher data . . . is given in advance,” and “is a highly accurate method for estimating [a] class” for a newly observed feature vector. Ex. 1006, [0056]-[0057]; *see id.*, [0031] (“The supervised machine learning system outlined above can be re-learned by providing feedback of new data each time an arbitrary configurable cycle arrives.”).

Mori explains that, in one example, 100 known malware files and 100 known “normal files” were used for machine learning. *Id.*, [0062]. Mori further notes that, “not only can we determine whether the subject is malware or a normal file, but also use machine learning to classify *other classes* of malware and determine the malware type.” *Id.*, [0063].

Wherein one or more features of the feature vector are selectively turned on or off 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. Mori discloses this portion of [1.e]. Ex. 1004 ¶¶ 285-93. Mori generates a feature vector for each Object based on a “word set” comprising the “printed and readable byte strings [extracted] as

words from the Object” using, for example, “a static dictionary provided in advance.” Ex. 1006, [0029]-[0032]. Mori explains that the feature vector can be defined “as whether a word existed or not by assigning it ‘0’ or ‘1.’” *Id.*, [0030]. In other words, features are selectively turned on or off based on whether or not a word in the predetermined dictionary is present in the Object or not. *See* Ex. 1004 ¶¶ 285-91.

Mori provides an example wherein the “[w]ord extract unit . . . uses word dictionary . . . prepared in advance for the word candidates shown in Table 2 above, *extracting only those words that match the words in [the] word dictionary.*” Ex. 1006, [0049]-[0050]. Mori further explains that, alternatively, “an Internet search engine is used to search for appropriate word candidates, with words extracted only when the number of search results *exceeds an arbitrary configurable threshold.*” *Id.*, [0050].

Mori discloses the use of “known tf-idf values” as a configuration example for the generated feature vectors. *Id.*, [0053]. A so-called “tf-idf value” (*i.e.*, term frequency-inverse document frequency) is a measure that reflects the importance or relevance of string representations in a single document amongst a collection (or corpus) of documents. Ex. 1004 ¶¶ 288. “Term frequency” refers to the number of times the word appears in a document, and “inverse document frequency” represents how common (or uncommon) a word is among the corpus. Multiplying these values

together results in tf-idf values that have higher scores for more important/relevant terms, and lower scores (approaching zero) as a term becomes less relevant. *Id.*; see Ex. 1006, [0053] (explaining that “known tf-idf values can be used as a configuration example of v_{ij} ,” or “ $v_{ij} = tf_{ij} * idf_{ij}$ ” for each word w_j in a file). Mori’s Table 4 depicts the “result of analyzing the words that appear in each file as well as the word counts for each file”:

Table 4

File name (hash value)	Appearance of word: number of times
00c4fc2fd0ac1d2b2fa43d8c5a087104	dll:4,get:2,sip:2,kernel:2,address:1,change:1,...
013a5ba10e3fc8a039b045530381d957	dll:4,get:2,kernel:2,rhd:1,bde:1,amu:1,range:1,...
notepad.exe	get:47>window:20,set:19,file:16,message:13,...
****	*****

Id., [0059], [Table 4]. Mori’s Table 5 depicts “an example of configuring feature vectors” based on Table 4, wherein each word is numbered based on the word dictionary:

Table 5

File ID	Feature Vector	Class
1	119:0.0008, 196:0.0077, 307:0.0077,...	-1
2	119:0.0024, 168:0.0051, 213:0.0382,...	-1
3	84:0.006, 94:0.002, 113:0.00258,...	+1
****	*****	****

Id., [0060], [Table 5]. As shown in Table 5, words that are present in each file are represented as “{word number: tf-idf value}” in the feature vector. *Id.*, [0060]. Thus, if a word is not present, the “static data point” for that word is “turned off,” since it is not included in the generated feature vector for that file (*i.e.*, the tf-idf value is 0 because the tf portion is 0). Ex. 1004 ¶ 291. Turning off a data point when the word is not present is based on whether a data point “is within a predetermined range” as required by this portion of [1.e], and consistent with the district court’s construction of “within a predetermined range.” Specifically, the tf-idf “range” has a lower limit of ‘0’ (and an unbounded upper limit, *i.e.*, $\text{tf-idf} > 0$). Any tf-idf value that exceeds the lower limit of ‘0’ does not turn off the “static data point” for that word. Ex. 1004 ¶¶ 291-93. Conversely, where a word’s tf-idf value is outside the $\text{tf-idf} > 0$ range (*i.e.*, it is exactly ‘0,’ meaning that it is not present in a file), its “static data point” is turned off. *Id.*

[1.f]

Mori discloses [1.f]. Ex. 1004 ¶¶ 294-98. Mori describes the use of “support vector machines (SVMs)” as a “widely used” and “known” supervised learning method for estimating a class from an observed feature vector. Ex. 1006, [0057]-[0058]. Mori’s Table 6 illustrates an “example of the results of applying SVM to the above feature vector,” and shows 221 files classified as malware, and 1526 files classified as “normal” files. *Id.*, [0061]-[0063]. One of skill in the art at the time of

the effective filing date of the '869 Patent would have understood Mori's disclosure of using and applying a "support vector machine (SVM)" to be synonymous with "using support vector processing" as required by [1.f], and the district court's claim construction of "support vector processing." *See* Ex. 1004 ¶¶ 294-96.

Moreover, a POSITA would have understood that the "support vector machines" described in Mori would have been used to build a probabilistic model to classify the files as required by [1.f], because a probabilistic model would be necessary to "not only determine[] whether or not a file contains malware or is a normal file, but also determines the type of malware by machine learning *that enables classification of other classes.*" Ex. 1006, [0033]; *see* Ex. 1004 ¶ 298. As explained above with respect to Ideses in Ground 1, SVMs are typically used as non-probabilistic binary classifiers, meaning that they are capable of classifying a new example into one of two classes. *Id.* However, a well-known technique called "Platt scaling" can be used to allow an SVM to predict a probability distribution over a set of classes, rather than between two. *Id.* It would have been obvious to a POSITA to use Platt scaling with the SVM technique described in Mori in order to allow the SVM to classify a file into "other classes of malware" beyond just determining that a file is malware. Ex. 1006, [0063] ("By using malware type c_k ($k=1, 2, \dots, K$) as a class and applying machine learning to a given feature for problem setting the determination of class C_k , such classification is possible."); Ex. 1004 ¶ 298.

b. Claim 2

Mori discloses [2]. Ex. 1004 ¶¶ 299-301. Mori’s malware detection method is performed “without decrypting or unpacking the executable file” as required by claim [2]. *Id.* Mori explains that “[m]alware often uses deliberately complex encryption and compression (called packing) to avoid detection by pattern matching or static analysis by inverse assemblers,” and that “[p]acked executable files decrypt themselves at runtime and are then deployed in memory.” Ex. 1006, [0003]. And, in general, “in order to perform a detailed analysis of malware, packed executable files need to be decrypted.” *Id.*, [0004]. Mori’s method is an effort to detect malware without using “decryption, disassembly, and debuggers,” as such techniques “are not suitable for use in detecting communications containing malware in real time.” *Id.*, [0007]. Accordingly, Mori focuses on the analysis of “readable strings contained in software files,” which “can collect a variety of information about malware.” *Id.*, [0023], [0025]; Ex. 1004 ¶¶ 300-01.

c. Claim 3

Mori discloses [3]. Ex. 1004 ¶¶ 302-05. As explained in [1.f], Mori discloses “providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.”

As explained above with respect to Ideses in Ground 1, a POSITA would have understood that “support vector machines” that use Platt scaling to build a probabilistic model would have utilized data analysis and pattern recognition to generate a feature vector. *See* Ex. 1004 ¶¶ 303-05. Platt scaling is used to “train the parameters of an additional sigmoid function to map the SVM outputs into probabilities,” and then the SVM+sigmoid combination is used to determine the probabilities that a given example belongs to each category. *See id.*; *see also* Ex. 1013, 1, 3-7. It would have thus been obvious to a POSITA that an SVM “builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors” when it is used with Platt scaling by fitting a two-parameter sigmoid function that maps classification probabilities to the SVM outputs (*i.e.*, the “feature vectors”). *See id.*; *see* Ex. 1004 ¶ 305.

d. Claim 4

Mori discloses [4]. Ex. 1004 ¶¶ 306-08. As explained in [1.f], Mori’s Table 6 illustrates an “example of the results of applying SVM to the above feature vector,” and shows 221 files classified as malware, and 1526 files classified as “normal” files. *Id.*, [0061]-[0063]. A POSITA would have understood that classifying “malware” indicates “whether the executable file is harmful” as required by claim [4]. Ex. 1004 ¶¶ 306-08.

e. Claim 5

Mori discloses and/or renders obvious [5]. Ex. 1004 ¶¶ 309-13. As explained above with respect to [1.c], Mori discloses “identifying an executable file to be evaluated.” Mori explains that in the first step of its malware detection method, “the file or communication data to be sent or received (hereinafter, the file or communication data is collectively referred to as the ‘Object’) is recorded by the Object recording unit” Ex. 1006, [0028]. When the object recording unit records the file in response to it being a file that is “to be sent or received,” “sent or received” is the claimed “detected condition” and the recording is the claimed “identified in response.”

f. Claim 6

Mori discloses and/or renders obvious [6]. Ex. 1004 ¶¶ 314-16. As explained above with respect to [1.c] and [5], Mori discloses that a file can be identified when it is to be “received.” A POSITA would understand, or it would be obvious, that a file that is to be received could be when a user requests for a file to be downloaded. *Id.*

g. Claim 8

Mori discloses [8]. Ex. 1004 ¶¶ 317-19. Mori explicitly states that its method “focuses on readable strings contained in software files,” which “include messages contained in programs that make up malware, host names used for communication, execution command names, function names, library names, and so on, and can

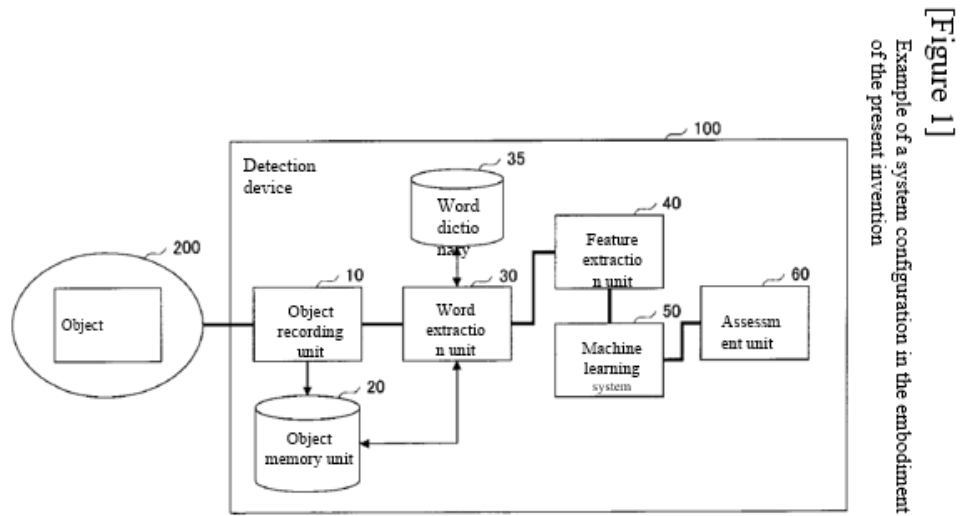
collect a variety of information about malware.” Ex. 1006, [0023], [0025]. Mori explains that its method is useful for detecting “[malware] variants where only part of the code has been modified,” because “*it is likely that the characteristics of the printable strings [in a variant] will be similar to the original.*” *Id.*, [0026]. Accordingly, Mori’s word extracting unit uses a “word dictionary . . . prepared in advance” and “extract[s] *only those words that match the words in word dictionary.*” *Id.*, [0050]. In other words, Mori discloses that the extracted words “represent predefined character strings in the executable” to the extent that they match the words in a word dictionary “prepared in advance,” that is, “predefined.” *See* Ex. 1004 ¶¶ 317-19.

h. Claim 9

Mori discloses [9]. Ex. 1004 ¶¶ 320-23. Mori explains that the “[m]achine learning system 50 can also perform re-learning via feedback of new data upon each arbitrary configurable cycle.” Ex. 1006, [0064]. Mori explains that such “new data” refers to “data about files that are not real-time, such as through prior art, or that, over time, have been determined to be malware or normal files.” *Id.*

i. Independent Claim 10

To the extent the preamble of claim 10 is limiting, Mori discloses [10.pre]. Ex. 1004 ¶¶ 325-27. Mori discloses “a system configuration,” as shown in Figure 1:



Ex. 1006, [0036], Fig. 1. Mori explains that “[d]etection device 100 is connected to network 200 and consists of Object recording unit 10, Object storage unit 20, word extraction unit 30, word dictionary 35, feature extract unit 40, machine learning system 50, and assessment unit 60,” and that “the Object storage unit 20 and word dictionary 35 are storage media such as a hard disk.” *Id.*, [0037].

[10.a]

See claim [1.a].

[10.b]

See claim [1.b].

[10.c]

See claim [1.c].

[10.d]

See claim [1.d].

[10.e]

See claim [1.e].

[10.f]

See claim [1.f].

j. Claim 11

See claim [2].

k. Claim 12

See claim [3].

l. Claim 13

See claim [4].

m. Claim 14

See claim [8].

n. Independent Claim 15

[15.pre]

See claim [1.pre].

[15.a]

See claim [1.a].

[15.b]

See claim [1.b].

[15.c]

See claim [1.c].

[15.d]

See claim [1.d].

[15.e]

See claim [1.e].

[15.f]

See claim [1.f].

o. Claim 16

See claim [2].

p. Claim 17

See claim [8].

q. Independent Claim 18

[18.pre]

See claim [1.pre].

[18.a]

See claim [1.a].

[18.b]

See claim [1.b].

[18.c]

See claim [1.c].

[18.d]

See claim [1.d].

[18.e]

See claim [1.e].

[18.f]

See claim [1.f].

r. Claim 19

See claim [2].

s. Claim 20

Mori discloses [20]. Ex. 1004 ¶¶ 356-57. As explained above in [1.f], Mori discloses using supervised learning to classify a file. Mori discloses that “a variety of known methods widely” can be used, including “neural networks.” Ex. 1006, [0057].

t. Claim 21

See claim [1.f].

u. Claim 22

Mori discloses [22]. Ex. 1004 ¶¶ 360-62. As explained above in [1.f], Mori discloses using supervised learning to classify a file. Mori discloses that “a variety of known methods widely” can be used including “decision trees.” Ex. 1006, [0057].

v. Claim 23

Mori discloses [23]. Ex. 1004 ¶¶ 363-65. As explained above in [1.f], Mori discloses using supervised learning to classify a file. Mori discloses that “a variety of known methods widely” can be used including “Bayesian methods.” Ex. 1006, [0057].

w. Claim 24

See claim [1.f].

x. Claim 26

See claim [5].

y. Claim 27

See claim [6].

z. Claim 29

See claim [8].

aa. Claim 30

See claim [9].

bb. Independent Claim 32

[32.pre]

See claim [1.pre].

[32.a]

See claim [1.a].

[32.b]

See claim [1.b].

[32.c]

See claim [1.c].

[32.d]

See claim [1.d].

[32.e]

See claim [1.e].

[32.f]

See claim [1.f].

cc. Claim 33

See claim [2].

dd. Claim 34

See claim [20].

ee. Claim 35

See claim [1.f].

ff. Claim 36

See claim [22].

gg. Claim 37

See claim [23].

hh. Claim 38

See claim [1.f].

ii. Claim 40

See claim [5].

jj. Claim 41

See claim [6].

kk. Claim 43

See claim [8].

ll. Claim 44

See claim [9].

mm. Independent Claim 46

[46.pre]

See claim [10.pre].

[46.a]

See claim [1.a].

[46.b]

See claim [1.b].

[46.c]

See claims [10.pre] and [1.c].

[46.d]

See claim [1.d].

[46.e]

See claim [1.e].

[46.f]

See claim [1.f].

nn. Claim 47

See claim [2].

oo. Claim 48

See claim [20].

pp. Claim 49

See claim [1.f].

qq. Claim 50

See claim [22].

rr. Claim 51

See claim [23].

ss. Claim 52

See claim [1.f].

tt. Claim 54

See claim [5].

uu. Claim 55

See claim [6].

vv. Claim 57

See claim [8].

ww. Claim 58

See claim [9].

C. Ground 3: Claims 1-59 Are Rendered Obvious Over Ideses in View of Mori

1. Overview of Ground

As demonstrated in Ground 1, Ideses discloses (or renders obvious) all of the limitations of the Challenged Claims, and as demonstrated in Ground 2, Mori renders obvious claims 1-6, 8-24, 26-27, 29-30, 32-38, 40-41, 43-44, 46-52, 54-55, and 57-58. Ex. 1004 ¶¶ 77-396. To the extent Patent Owner argues Ideses does not sufficiently teach or suggest any element of any of the challenged independent claims, Mori discloses these features. Thus, the combination of Ideses and Mori renders obvious claims 1-59 of the '869 Patent. Ex. 1004 ¶¶ 397-408.

2. Analysis of Unpatentability and Motivation to Combine

It would have been obvious to a POSITA at the effective filing date of the '869 Patent to have combined Mori's malware analysis techniques with Ideses' malware detection methods. Ex. 1004 ¶¶ 397-408. A POSITA reading Ideses would have been motivated to incorporate Mori's teachings to analyze encrypted and/or packed executable files without decrypting or unpacking those files as described in Mori. *Id.* Both Ideses and Mori relate to classifying and detecting malware. Ex. 1005, Abstract, [0123]; Ex. 1006, [0010], [0016]. Ideses and Mori both relate to analyzing files by extracting features from them. Ex. 1005, [0059]-[0060]; Ex. 1006, [0007], [0023]. A POSITA would have been motivated to incorporate Mori's

teachings about classifying and detecting malware by extracting features when reading Ideses. Ex. 1004 ¶¶ 397-408.

Petitioners incorporate by reference the analysis of the Ideses and Mori grounds above. *See* Sections IX.A.2 and IX.B.2, *supra*.

Extracting . . . without decrypting or unpacking the executable file (elements [2], [11], [16], [19], [33], [47]). As explained, Mori discloses a malware detection technique that “focuses on readable strings contained in software files,” distinguishable from other techniques that require decrypting or unpacking the executable file, which “are not suitable for use in detecting communications containing malware in real time.” Ex. 1006, [0007], [0023]. In the “Problem to be Solved by the Invention” section, Mori notes that prior art malware detection methods “that use decryption, disassembly, and debuggers are time-consuming and require computational resources.” *Id.*, [0007]. Mori identifies “an approach to detecting whether malware is present by executing it” as “time-consuming and requiring computational resources, meaning that timely detection and determination could not be performed.” *Id.*, [0008]; Ex. 1004 ¶¶ 400-04.

Ideses likewise recognizes that “resource limited” mobile devices may be “unable to extract the full set of [classification] features during runtime, and/or unable to perform the extraction within a reasonable amount of time to allow for run-time operation.” Ex. 1005, [0060]. Thus, a POSITA would have been motivated

to combine the Ideses method with Mori’s “readable strings” technique, which is an offline (*i.e.*, static analysis) technique, in order to enable a resource-limited device to perform feature extraction and malware analysis locally. Ex. 1004 ¶¶ 400-04.

A POSITA would have had a reasonable expectation of success, because both Ideses and Mori describe offline/static malware analysis techniques. *E.g.*, Ex. 1005, [0090] (“The feature extraction may take place off-line, not part of a run-time operation.”); Ex. 1006, [0008]-[0009] (distinguishing “an approach to detecting whether malware is present by executing it” from the “present invention”). Moreover, the combination would have been nothing more than the use of a known technique (Mori’s focus on “readable strings” for offline malware detection) to improve similar devices (Ideses’ malware detection system) to yield predictable results (the ability to detect malware without decrypting or unpacking an executable). Ex. 1004 ¶¶ 400-04.

Predefined character strings ([8], [14], [17], [29], [43], [57]). As explained above, Mori “focuses on *readable strings contained in software files*, collects information (characteristic vectors) that characterizes malware, such as messages contained in programs that constitute malware,” and “compares this content with strings contained in files.” Ex. 1006, [0023]. Accordingly, Mori explicitly discloses the extraction of “predefined character strings” as required by these claim elements. *See* Section IX.B.2.a, *supra*; Ex. 1004 ¶¶ 405-08.

Ideses likewise explains that “the extracted features may contain *static meta data* regarding the application, for example . . . name,” *Id.*, [0089]. A POSITA would have been motivated to use Mori’s examples of predefined “readable strings . . . such as messages, . . . host names, execution command names, function names and library names,” Ex. 1006, [0023], as additional features for extraction in the Ideses method, for the same reasons described above. Ex. 1004 ¶¶ 405-08.

D. Ground 4: Claims 1-59 Are Indefinite Under 35 U.S.C. § 112(b)

Claims 1-59 are indefinite because the claims are not “precise enough to afford clear notice of what is claimed.” *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 909 (2014) (citation omitted); Ex. 1004 ¶¶ 397-408. The boundaries of independent claims 1, 10, 15, 18, 32, and 46 are unclear, because the term “the static data points” lacks a reasonably certain antecedent basis. After reciting different types of static data points, the claim language recites “*the* static data points” without providing clarity as to whether “the static data points” refers to the “identified” or “plurality of” static data points. Because a POSITA could not ascertain the boundaries of “the static data points” and therefore would not know how to avoid infringement, the term is indefinite. *Id.* Each of the dependent claims depends from one of independent claims 1, 10, 15, 18, 32, and 46 and is thus also invalid.

1. Legal Standard

Claim language is indefinite where several meanings are possible. *Bushnell Hawthorne, LLC v. Cisco Sys. Inc.*, 813 F. App'x 522, 526-27 (Fed. Cir. 2020) (finding term “said different IP Address” indefinite when its claim described three different IP addresses). Even under the former “insolubly ambiguous” standard, courts routinely deemed indefinite limitations without an antecedent basis. *E.g., In re Packard*, 751 F.3d 1307, 1309-10 (Fed. Cir. 2014) (affirming PTAB’s rejection of claims as indefinite where limitations “lacked an antecedent basis or were otherwise unclear”). The MPEP further confirms that a claim is unclear when it has an unclear antecedent basis. MPEP 2173.05 (“[I]f two different levers are recited earlier in the claim, the recitation of ‘said lever’ in the same or subsequent claim would be unclear where it is uncertain which of the two levers was intended.”).

2. The Independent Claims Have Unclear Antecedent Basis

The asserted claims of the '869 Patent recite “identified static data points” (green below) and “plurality of static data points” (blue below):

1. A computer-implemented method comprising:

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

associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;

...

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

generating a feature vector from the plurality of static data points using a classifier trained to classify *the static data points* based on training data, . . . wherein one or more features of the feature vector are selectively turned on or off 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

In the “generating” limitation, the claim recites “the static data points” (red), without indicating whether it refers to the “identified static data points” or the “plurality of static data points.” For example, it is not clear whether the “classifier trained to classify the static data points” is referring to “the identified static data points” which were associated with categories of files or “the plurality of static data points” which have been used to generate a feature vector. The specification provides no guidance on how to resolve the ambiguity created in the claim language. Because “the static data points” could refer to either the “identified” or “plurality of” static data points, the lack of clarity makes the claims indefinite.

E. Grounds 5 and 6: Claims 8, 14, 17, 29, 43, and 57 Are Indefinite Under 35 U.S.C. § 112(b) and Lack Written Description

Claims 8, 14, 17, 29, 43, and 57 (“string claims”) are indefinite, because the claims are not “precise enough to afford clear notice of what is claimed.” *Nautilus*, 572 U.S. at 909 (citation omitted); Ex. 1004 ¶¶ 413-25. Specifically, the string claims require the “static data points” extracted from the executable file to be

character strings, but the independent claims on which they rely require those same data points to have a numeric “value” that can be “within a predetermined range.” A character string does not have a numerical value; thus, the string claims are internally inconsistent and nonsensical, and cannot provide reasonable certainty about the scope of the invention.

The '869 Patent distinguishes between different types of information that can be extracted from an executable file for analysis. Table 1.1, for example, provides a list of different static data points and organizes them into distinct categories (*e.g.*, “Numeric values” in the first column, “Nominal values” in the second column, “Strings/Byte sequences” in the third column, and “Boolean values” in the fourth column):

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 heap	file OS	special build	Code Characteristics Anomaly
size of stack	file type	product version	Code Name Anomaly
size of image	machine type	file version	Count Anomaly
PE header time	PE type	package code	Data Characteristics Anomaly
Section Entropy	section counts	product code	Data Name Anomaly
Sections count	DLL count	export DLL name	Export Exception
	DLL functions	assembly version	Large Number of DLLs Anomaly
	data directory	Certificate Issuer	flag DLL Name Anomaly
	export count	Certificate 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

The string claims are expressly limited to extracting character strings, or static data points representing character strings, from the executable file (yellow highlighting below). The independent claims upon which they rely then require generating a feature vector from the character strings. Portions of the feature vector are then selectively turned on or off based at least in part on whether the “value” of the extracted character strings is within a predetermined range (blue highlighting below). As an example, independent claim 18 (excerpts) and dependent claim 29 are shown below:

Claim 18 (excerpts)	Claim 29
... extracting a plurality of static data points from the executable file; generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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 ...	29. The computer-implemented method according to claim 18, wherein the plurality of static data points represent predefined character strings in the executable file.

The problem is that the “predetermined range” used in the context of the ’869 Patent and these specific claim limitations refers to a numeric range. There is only

one example in the '869 Patent of determining whether an extracted static data point is within a predetermined range:

For example, ranges of data can be identified for static data points that may enable the learning classifier to controllably select (e.g., turn on/off) features for evaluation by learning classifier. In one instance ***where a static data point extracted for evaluation is a file size (e.g., numeric value), the learning classifier may turn on/off features for evaluation based on whether [the] file size of the file is within a certain range.***

Ex. 1001, 7:45-52. This sole example describes determining whether a numeric static data point (file size) is within a predetermined range. There is nothing in the '869 Patent disclosing non-numeric ranges that can be used to decide whether features in the feature vector are turned on or off. In fact, when the '869 Patent discusses analyzing other non-numeric information (such as “legal information”) to determine whether to turn on/off features for evaluation, the patent only discloses detecting whether or not the file contains the non-numeric information, not whether the non-numeric information somehow falls within a predetermined range. *Id.*, 7:53-7:64.

Because the claims require the extracted data points to be both non-numeric (character strings) and numeric (having a value that can be compared against a range) at the same time, they are internally inconsistent, rendering the claims indefinite. *Trs. of Columbia Univ. v. Symantec Corp.*, 811 F.3d 1359, 1366-67 (Fed. Cir. 2016) (holding claims indefinite because they “describe[d] the step of extracting machine

code instructions from something that does not have machine code instructions,” which was “nonsensical”); *Competitive Techs., Inc. v. Fujitsu Ltd.*, 185 F. App’x 958, 965-66 (Fed. Cir. 2006) (holding that a claim was internally inconsistent and therefore indefinite, because the “address means” limitation of the claim required ISA structures, and the “sustain means” limitation of the same claim excluded ISA structures); *Koki Holdings Co. v. Kyocera Senco Indus. Tools, Inc.*, 2021 WL 1092579, at *1 (D. Del. Mar. 22, 2021) (“A claim that is nonsensical or requires an impossibility is indefinite as a matter of law”); *Fundamental Innovation Sys. Int’l LLC v. Samsung Elecs. Co.*, 2018 WL 647734, at *40 (E.D. Tex. Jan. 31, 2018) (“Claim 2 ... is therefore internally inconsistent and, as a result, is indefinite.”).

While computers ultimately represent information using binary numbers, the claims and the specification disclose static data points, “strings,” as textual information (e.g, “company name,” “file description,” “internal name”) that are different from “numeric values,” which are disclosed as a separate category of data points. Ex. 1001, Table 1.1 (distinguishing between “numeric values” in the first column and “strings” in the third column). The ’869 Patent does not explain how such a “value” should be computed from strings so that it could be measured against a “range.” The claims cannot be re-written to give “character strings” a meaning contradicted by the specification and the claim language itself. *Haemonetics Corp.*

v. Baxter Healthcare Corp., 607 F.3d 776, 782 (Fed. Cir. 2010) (“[W]e do not redraft claims to contradict their plain language in order to avoid a nonsensical result.”).

The ’869 Patent also does not provide for a non-numeric “predetermined range.” For example, the different categories of textual information disclosed in the specification (*e.g.*, “legal information”) cannot constitute a “range,” because the specification does not use the term “range” to refer to non-numeric categories. Ex. 1001, 7:53-7:65. The ’869 Patent uses the words “categories” and “range” differently throughout its specification and claims. (*Compare id.*, 5:67-6:5, 6:17-19, Table 1.1 *with id.* at 7:45-53.) A POSITA would not understand “predetermined range” as recited in the claims to refer to “predetermined categories.” Ex. 1004 ¶¶ 413-25. The patent’s *only* example of determining whether an extracted static data point is within a predetermined range concerns a “file size,” which is an actual numeric value.

Under the governing *Nautilus* standard, it is not enough that a POSITA “can ascribe some meaning to a patent’s claims.” *Interval Licensing LLC v. AOL, Inc.*, 766 F.3d 1364, 1370-71 (Fed. Cir. 2014). Instead, “[t]he claims, . . . must provide objective boundaries for those of skill in the art.” *Id.* at 1371. Here, there is no objective standard in the claims, specification, or prosecution history explaining how the claimed data points can be character strings and numeric values at the same time, as required by the claim language, nor is there any objective standard suggesting

how a POSITA could determine whether values of character strings could be within a range. Instead, the plain language of the string claims, “read in light of the specification delineating the patent, and the prosecution history,” requires a nonsensical impossibility: that the data points are both character strings and numeric values at the same time. The claims are therefore indefinite. *Synchronoss Techs., Inc. v. Dropbox, Inc.*, 987 F.3d 1358, 1366-67 (holding that claims reciting “generating a [single] digital media file” that itself “compris[es] a directory of digital media files” were indefinite as a matter of law, because the claims were nonsensical and required an impossibility).

As discussed, there is no written description in the specification of how character strings can fall within a numeric “range.” Ex. 1004 ¶¶ 413-25. There is also no disclosure, of non-numeric ranges that can be used to decide whether features in the feature vector are turned on or off. *Id.* As a result, the string claims are also invalid for lack of written description. *Transperfect Glob., Inc. v. Matal*, 703 F. App’x 953, 963 (Fed. Cir. 2017) (finding lack of written description when “specification provide[d] no disclosure”).

X. PTAB DISCRETION SHOULD NOT PRECLUDE INSTITUTION

A. PTAB Should Not Exercise Its Discretion to Deny Institution Under *Fintiv*

Consistent with congressional intent and the goals of *NHK/Fintiv*, Petitioners took all possible steps to promote the Board’s efficiency and fairness objectives.

Petitioners initiated this petition with diligence, filed a single, focused petition, and served a stipulation (Ex. 1003) broader than one the Board previously found which had mitigated concerns of duplicative efforts between the district court and the Board and had potentially conflicting decisions. *Sand Revolution II, LLC v. Continental Intermodal Grp.*, IPR2019-01393, Paper 24 (June 16, 2020). These facts, along with the merits of the Grounds, strongly favor institution.

1. Factor 1: Likelihood of Stay

The District Court has made clear in prior cases that a motion to stay prior to institution would be premature, and the Board recently “decline[d] to speculate on the likelihood of Judge Albright entering a stay after institution” and thus found Factor 1 neutral. *Western Digital Corp. v. Kuster*, IPR2020-01391, Paper 10 at 9 (Feb. 16, 2021); *see also Volkswagen Grp. of Am., Inc. v. Arigna Tech. Ltd.*, IPR2021-01321, Paper X at 3 (Feb. 15, 2022) (declining to speculate about the likelihood of a stay). Thus, this factor is at worst, neutral.

2. Factor 2: District Court Schedule

The District Court schedule weighs in favor of institution. The Litigation includes more than 20 patents and five separate lawsuits that are consolidated for pre-trial purposes. Pursuant to the District Court’s amended scheduling order on January 22, 2022 (Dkt. No. 160), all trials are set for August 19, 2024. Ex. 1008, 6. Based on an 18-month PGR schedule, a Final Written Decision (“FWD”) would

issue around November 2024. But District Court schedules are uncertain as a general matter, and here there is even greater than usual uncertainty about when the challenged patent will be tried, weighing against discretionary denial.

Not only are there five separate litigations scheduled for the August 2024 trial date, but the number of patents in each case will almost certainly require each case to have multiple trials. The '869 Patent was added by amendment, and it is not certain to be addressed in the first trial, even if Petitioner is in the first trial. The Board has recognized that “multiple proceedings with trial dates close to” the same trial date, as here, counsels against discretionary denial. *Cizion, LLC v. Kerr Machine Co.*, PGR2020-00065, Paper 10 at 26 (Dec. 3, 2020).

Indeed, because projected trial dates are uncertain, the Board has held this factor “neutral,” even where the projected trial date was months before the Board’s statutory final-decision deadline. *Micron Tech., Inc. v. Godo Kaisha IP Bridge 1*, IPR2020-01008, Paper 10 at 14 (Dec. 7, 2020) (“Accordingly, even if we consider the Texas court to have set a formal trial date for a few months before the final written decision, due to the uncertainty as to this trial date, this factor is, at most, neutral to whether we should exercise our discretion to deny the Petition.”). In *Cizion*, trial was set “three months prior to [the Board’s] (anticipated) deadline for a final written decision.” PGR2020-00065, Paper 10 at 26 (Dec. 3, 2020). The Board explained that in spite of the scheduled trial date, evidence showed “the actual trial

date is uncertain.” *Id.* Relevant here, the *Cizion* Board evidence that “in the Western District of Texas, 70% of trial dates have slipped following PTAB discretionary denials.” *Id.* Indeed, recent Interim Guidance² dictates that because scheduled trial dates “are unreliable and often change,” a scheduled trial date “is not by itself a good indicator of whether the district court trial will occur before the statutory deadline for a [FWD].” *Samsung Elec. ’s Co., Ltd. et al v. Staton Techniya, LLC*, IPR2022-00302, Paper 13, 10 (PTAB July 11, 2022).

Finally, trial is more likely to get pushed further back in this case, because the litigation is especially complex. The District Court has stated this case is “about as complicated as you’re going to get with patent cases” and said it would be “amenable to adjusting the schedule” by “granting more time to address things.” Ex. 1019, 2. Because this multi-defendant consolidated case is exceptionally complicated, it is unlikely that the District Court will dispose of the dispute over the ’869 Patent before a FWD would issue.

Thus, this factor weighs strongly against discretionary denial.

3. Factor 3: Petitioners’ Investment in PGR Outweighs Forced Investment in Litigation to Date

Petitioners’ substantial investment in these PGR proceedings should counterbalance—if not outweigh—the resources invested in the co-pending

² https://www.uspto.gov/sites/default/files/documents/interim_proc_discretionary_denials_aia_parallel_district_court_litigation_memo_20220621_.pdf

litigation. It would be unjust to consider resources expended in the District Court (equally by both parties), without considering Petitioners' resources expended to prepare this PGR petition that would be irretrievably lost without consideration on the merits, in addition to the extensive expenses that that will certainly follow in the Litigation if this petition is not instituted. *See, e.g., Apple v. Seven Networks*, IPR2020-00255, Paper 13 at 12-15 (July 28, 2020) (declining to exercise discretionary denial on petitions filed nine months after the commencement of co-pending litigation where a large number of patents were included in the litigation, and a large number of claims from each patent were challenged in the petitions).

In *Sand Revolution*, even after invalidity contentions and the *Markman* hearing, the Board found that this factor weighed "only marginally, if at all, in favor of exercising discretion." IPR2019-01393, Paper 24, at 11. Given Petitioners' diligence and significant investment in filing this petition, this factor should weigh against a discretionary denial.

4. Factor 4: The Petition Raises Unique Issues

Factor 4 strongly supports institution. Consistent with *Fintiv*, Petitioners ask the Board to consider the unique challenges raised in the petition. Petitioners have stipulated that, if the Board institutes the present petition, Petitioners will not assert that the challenged claims are invalid on the instituted grounds in the related litigation, nor any other possible prior art printed-publication grounds based on the

references in Grounds 1-3. *Webroot, Inc. et al. v. Sophos Ltd.*, No. 6:22-cv-00240-ADA-DTG (W.D. Tex.). This stipulation is significantly broader than the stipulation in *Sand Revolution*, and therefore this factor strongly supports institution. *Sand Revolution*, IPR2019-01393, Paper 24 at 11; *see also Apple, Inc. v. AliveCor, Inc.*, IPR2021-00970, Paper 10 (Dec. 8, 2021).

5. Factor 5: Institution Will Provide the Board an Opportunity to Invalidate Claims That Later Could Be Reasserted Against Others

In the litigation, Patent Owner has asserted 15 patents against five defendants. Four patents are asserted against all five defendants (including the '869 Patent); several defendants have counterclaimed with patents. *Webroot, Inc. v. AO Kaspersky Lab*, No. 6:22-cv-00243-ADA-DTG (W.D. Tex.). Thus, Patent Owner has a demonstrated history of asserting claims against multiple defendants, and the Board has an opportunity to invalidate some of those claims here. Factor 5 weighs in favor of institution.

6. Factor 6: Other Circumstances Support Institution

The merits of the unpatentability grounds are particularly strong, weighing in favor of institution. *Apple Inc. v. Fintiv, Inc.*, IPR 2020-00019, Paper 15 at 14-15 (May 13, 2020). And the effect on “the economy [and] the integrity of the patent system” further supports institution. PTAB Consolidated Trial Practice Guide, 56. In addition, the '869 Patent was issued in August and Petitioners have worked

expeditiously to prepare this Petition. The claims of the '869 Patent are very broad and purport to cover extracting data from an executable file and using machine learning to detect malware, which was already in disclosed several years prior to the effective filing date of the patent. The Board should consider and invalidate the claims so that they may not be asserted against other entities in the future.

Moreover, if the Board institutes Petitioners' '844 Patent Petition, it should also institute the present petition since the claims are so similar and issues should be resolved consistently.

**B. PTAB Should Not Exercise its Discretion To Deny Institution
Under *Becton* and *Advanced Bionics***

Institution should not be denied under the framework set forth in *Becton, Dickinson & Co. v. B. Braun Melsungen AG*, IPR2017-01586, Paper 8 (Dec. 15, 2017) (precedential as to § III.C.5, first paragraph) and *Advanced Bionics LLC v. MED-EL Elektromedizinische Gerate GmbH*, IPR2019-01469, Paper 6 (PTAB Feb. 13, 2020) (precedential). Neither Ideses nor Mori was before the Examiner during prosecution. Therefore, the arguments presented in this Petition are not the same as, or substantially similar to, arguments previously presented to the Office. Because the first prong of the *Advanced Bionics* framework is not met, there is no need to reach the second prong to resolve against discretionary denial under Section 325(d)—though in any event, there are no pending proceedings before the PTAB concerning the '869 Patent, so the second prong is not met either. *Oticon Medical*

AB v. Cochlear Ltd., IPR2019-00975 Paper 15, 20 (PTAB Oct. 16, 2019)
(precedential).

XI. CONCLUSION

Petitioners have established a reasonable likelihood that the Challenged Claims are unpatentable. Petitioners therefore respectfully request that PGR be instituted.

DATED: May 8, 2023

Respectfully Submitted,

By: /s/ Benjamin Hershkowitz

Benjamin Hershkowitz
(Reg. No. 41,833)
200 Park Avenue
New York, NY 10166
Telephone: 212.351.2410
Facsimile: 212.351.6210
bhershkowitz@gibsondunn.com

*Attorney for Petitioners Sophos Ltd. and
Sophos Inc.*

CERTIFICATION UNDER 37 C.F.R. § 42.24(d)

Under the provisions of 37 C.F.R. § 42.24(a)(1)(ii), the undersigned attorney hereby certifies that the word count for Sections I and III–XI of the foregoing Petition for PGR is 18,503, according to the word count tool in Microsoft Word.

DATED: May 8, 2023

By: /s/ Benjamin Hershkowitz

Benjamin Hershkowitz
(Reg. No. 41,833)
200 Park Avenue
New York, NY 10166
Telephone: 212.351.2410
Facsimile: 212.351.6210
bhershkowitz@gibsondunn.com

*Attorney for Petitioners Sophos Ltd. and
Sophos Inc.*

CERTIFICATE OF SERVICE

The undersigned certifies service on the Patent Owner via FedEx overnight mail of a copy of this Petition for *Inter Partes* Review and supporting materials at the correspondence address of record for the '869 Patent shown in USPTO Patent Center:

Sprinkle IP Law Group
1301 W. 25th Street, Suite 400
Austin, TX 78705

Additionally, a courtesy copy has been mailed to Patent Owner at:

Webroot Inc.
385 Interlocken Crescent, Suite 800
Broomfield, CO 80021

DATED: May 8, 2023

By: /s/ Benjamin Hershkowitz

Benjamin Hershkowitz
(Reg. No. 41,833)
200 Park Avenue
New York, NY 10166
Telephone: 212.351.2410
Facsimile: 212.351.6210
bhershkowitz@gibsondunn.com

*Attorney for Petitioners Sophos Ltd. and
Sophos Inc.*

APPENDIX: CHALLENGED CLAIM LISTING

No.	Limitation
[1.pre]	A computer-implemented method comprising:
[1.a]	identifying, by a knowledge module, static data points that may be indicative of either a harmful or benign executable file;
[1.b]	associating, by the knowledge module, the identified static data points with one of a plurality of categories of files, the plurality of categories of files including harmful files and benign files;
[1.c]	identifying an executable file to be evaluated;
[1.d]	extracting, by the knowledge module, a plurality of static data points from the identified executable file;
[1.e]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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
[1.f]	providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.
[2]	The computer-implemented method according to claim 1, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[3]	The computer-implemented method according to claim 1, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.
[4]	The computer-implemented method according to claim 1, wherein the probabilistic model indicates whether the executable file is harmful.
[5]	The computer-implemented method according to claim 1, wherein the executable file is identified in response to a detected condition.

No.	Limitation
[6]	The computer-implemented method according to claim 5, wherein the detected condition is user request for a file download.
[7]	The computer-implemented method according to claim 5, wherein the detected condition is the detection of a new file attempting to execute.
[8]	The computer-implemented method according to claim 1, wherein the plurality of static data points represent predefined character strings in the executable file.
[9]	The computer-implemented method according to claim 1, wherein a determination of whether the executable file is harmful is used to retrain the classifier.
[10.pre]	A system comprising: at least one memory; and at least one processor operatively connected with the memory and configured to perform operation of:
[10.a]	identifying static data points that may be indicative of either a harmful or benign executable file;
[10.b]	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;
[10.c]	identifying an executable file to be evaluated;
[10.d]	extracting a plurality of static data points from the identified executable file; and
[10.e]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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
[10.f]	providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

No.	Limitation
[11]	The system according to claim 10, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[12]	The system according to claim 10, wherein the one or more support vector machines builds the probabilistic model by performing data analysis and pattern recognition on the one or more feature vectors.
[13]	The system according to claim 10, wherein the probabilistic model indicates whether the executable file is harmful.
[14]	The system according to claim 10, wherein the plurality of static data points represent predefined character strings in the executable file.
[15.pre]	A computer-readable storage device containing instructions, that when executed on at least one processor, causing the processor to execute a process comprising:
[15.a]	identifying static data points that may be indicative of either a harmful or benign executable file;
[15.b]	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;
[15.c]	identifying an executable file to be evaluated;
[15.d]	extracting a plurality of static data points from the identified executable file;
[15.e]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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
[15.f]	providing the generated feature vector to one or more support vector machines to build a probabilistic model that indicates whether the executable file fits into one of the categories of files.

No.	Limitation
[16]	The computer-readable storage device according to claim 15, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[17]	The computer-readable storage device according to claim 15, wherein the plurality of static data points represent predefined character strings in the executable file.
[18.pre]	A computer-implemented method comprising:
[18.a]	identifying static data points that may be indicative of either a harmful or benign executable file;
[18.b]	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;
[18.c]	identifying an executable file to be evaluated;
[18.d]	extracting a plurality of static data points from the executable file;
[18.d]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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
[18.e]	evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.
[18.f]	identifying static data points that may be indicative of either a harmful or benign executable file;
[19]	The computer-implemented method according to claim 18, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[20]	The computer-implemented method according to claim 18, wherein the machine learning model comprises an artificial neural network.
[21]	The computer-implemented method according to claim 18, wherein the machine learning model comprises a support vector machine.

No.	Limitation
[22]	The computer-implemented method according to claim 18, wherein the machine learning model comprises a machine learning decision tree.
[23]	The computer-implemented method according to claim 18, wherein the machine learning model comprises a Bayesian network.
[24]	The computer-implemented method according to claim 18, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.
[25]	The computer-implemented method according to claim 18, wherein the machine learning model comprises a clustering model.
[26]	The computer-implemented method according to claim 18, wherein the executable file is identified in response to a detected condition.
[27]	The computer-implemented method according to claim 26, wherein the detected condition is user request for a file download.
[28]	The computer-implemented method according to claim 26, wherein the detected condition is the detection of a new file attempting to execute.
[29]	The computer-implemented method according to claim 18, wherein the plurality of static data points represent predefined character strings in the executable file.
[30]	The computer-implemented method according to claim 18, wherein a determination of whether the executable file is harmful is used to retrain the classifier.
[31]	The computer-implemented method according to claim 18, further comprising preventing execution of the executable file based on classifying the executable file as harmful.
[32.pre]	A computer-readable storage device containing instructions, that when executed on a processor, cause the processor to execute a process comprising:
[32.a]	identifying static data points that may be indicative of either a harmful or benign executable file;

No.	Limitation
[32.b]	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;
[32.c]	identifying an executable file to be evaluated;
[32.d]	extracting a plurality of static data points from the executable file;
[32.e]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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
[32.f]	evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.
[33]	The computer-readable storage device according to claim 32, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[34]	The computer-readable storage device according to claim 32, wherein the machine learning model comprises an artificial neural network.
[35]	The computer-readable storage device according to claim 32, wherein the machine learning model comprises a support vector machine.
[36]	The computer-readable storage device according to claim 32, wherein the machine learning model comprises a machine learning decision tree.
[37]	The computer-readable storage device according to claim 32, wherein the machine learning model comprises a Bayesian network.
[38]	The computer-readable storage device according to claim 32, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.
[39]	The computer-readable storage device according to claim 32, wherein the machine learning model comprises a clustering model.

No.	Limitation
[40]	The computer-readable storage device according to claim 32, wherein the executable file is identified in response to a detected condition.
[41]	The computer-readable storage device according to claim 40, wherein the detected condition is user request for a file download.
[42]	The computer-readable storage device according to claim 40, wherein the detected condition is the detection of a new file attempting to execute.
[43]	The computer-readable storage device according to claim 32, wherein the plurality of static data points represent predefined character strings in the executable file.
[44]	The computer-readable storage device according to claim 32, wherein a determination of whether the executable file is harmful is used to retrain the classifier.
[45]	The computer-readable storage device according to claim 32, wherein the process further comprises preventing execution of the executable file based on classifying the executable file as harmful.
[46.pre]	A system comprising: at least one memory; at least one processor configured to perform operation of:
[46.a]	identifying static data points that may be indicative of either a harmful or benign executable file; and
[46.b]	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
[46.c]	at least one processor configured to perform operation of: identifying an executable file to be evaluated;
[46.d]	extracting a plurality of static data points from the executable file;
[46.e]	generating a feature vector from the plurality of static data points using a classifier trained to classify the static data points based on training data, the training data comprising files known to fit into one of the plurality of categories of files, wherein one or more features of the feature vector are selectively turned on or off based 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

No.	Limitation
[46.f]	evaluating the feature vector using a machine learning model to determine whether the executable file fits into one of the categories of files.
[47]	47. The system according to claim 46, wherein the plurality of static data points are extracted without decrypting or unpacking the executable file.
[48]	48. The system according to claim 46, wherein the machine learning model comprises an artificial neural network.
[49]	49. The system according to claim 46, wherein the machine learning model comprises a support vector machine.
[50]	50. The system according to claim 46, wherein the machine learning model comprises a machine learning decision tree.
[51]	51. The system according to claim 46, wherein the machine learning model comprises a Bayesian network.
[52]	52. The system according to claim 46, wherein evaluating the feature vector using the machine learning model comprises evaluating the feature vector using support vector processing.
[53]	53. The system according to claim 46, wherein the machine learning model comprises a clustering model.
[54]	54. The system according to claim 46, wherein the executable file is identified in response to a detected condition.
[55]	55. The system according to claim 54, wherein the detected condition is user request for a file download.
[56]	56. The system according to claim 54, wherein the detected condition is the detection of a new file attempting to execute.
[57]	57. The system according to claim 46, wherein the plurality of static data points represent predefined character strings in the executable file.
[58]	58. The system according to claim 46, wherein a determination of whether the executable file is harmful is used to retrain the classifier.

No.	Limitation
[59]	The system according to claim 46, further comprising at least one processor configured to perform operation of: preventing execution of the executable file based on classifying the executable file as harmful.