

Filed on behalf of: Wiz, Inc.

By: Matthew A. Argenti (margenti@wsgr.com)
Michael T. Rosato (mrosato@wsgr.com)
Wesley E. Derryberry (wderryberry@wsgr.com)
Tasha M. Thomas (tthomas@wsgr.com)
Joseph M. Baillargeon (jbaillargeon@wsgr.com)
WILSON SONSINI GOODRICH & ROSATI
650 Page Mill Road
Palo Alto, CA 94304

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

WIZ, INC.,
Petitioner,

v.

ORCA SECURITY LTD.,
Patent Owner.

Case IPR2025-00095
Patent No. 11,637,855

**PETITION FOR *INTER PARTES* REVIEW
OF U.S. PATENT NO. 11,637,855**

TABLE OF CONTENTS

I.	INTRODUCTION	1
II.	MANDATORY NOTICES UNDER 37 C.F.R. §42.8.....	2
III.	CERTIFICATIONS.....	3
IV.	IDENTIFICATION OF CHALLENGE; STATEMENT OF PRECISE RELIEF REQUESTED	3
V.	THE '855 PATENT.....	4
	A. Prosecution History	5
VI.	NO BASIS EXISTS FOR DENIAL UNDER 35 U.S.C. §325(D)	7
VII.	LEVEL OF ORDINARY SKILL	8
VIII.	CLAIM CONSTRUCTION	9
IX.	BACKGROUND	9
	A. Virtualization and Cloud Computing.....	9
	B. Cyber Security	10
X.	PRIOR ART.....	15
	A. Elder (U.S. Pub. No. 2014/0189873, EX1005).....	15
	B. Kim (U.S. Pub. No. 2016/0092679, EX1006)	16
	C. Hufsmith (U.S. Pub. No. 2020/0097662, EX1007)	17
XI.	GROUND 1: CLAIMS 1-6, 8-9, AND 11-18 WERE OBVIOUS OVER ELDER AND KIM.....	19
	A. Reasons to Combine Elder and Kim	20
	B. Independent Claims	27
	1. Preambles	27
	2. Element 1.i	28
	3. Elements 1.1, 12.1, and 18.1	29
	4. Elements 1.2, 12.2, and 18.2.....	32
	5. Elements 1.3, 12.3, and 18.3.....	33
	6. Elements 1.4, 12.4, and 18.4.....	36

7.	Elements 1.5 and 12.5.....	37
8.	Elements 1.6, 12.6, and 18.5.....	38
C.	Dependent Claims	45
1.	Claims 2-3 and 13-14.....	45
2.	Claims 4 and 15.....	46
3.	Claims 5 and 16.....	49
4.	Claims 6 and 17.....	50
5.	Claim 8.....	51
6.	Claim 9.....	52
7.	Claim 11	52
XII.	GROUND 2: CLAIMS 1-18 WERE OBVIOUS OVER ELDER, KIM, AND HUFSMITH	53
A.	Reasons to Combine Elder, Kim, and Hufsmith.....	53
B.	Claims 1, 12, and 18.....	59
C.	Claims 4 and 5.....	60
D.	Claim 7	62
E.	Claim 10	63
XIII.	CONCLUSION.....	65

LISTING OF CHALLENGED CLAIMS

1. A cyber security system for a cloud environment, the system comprising:

[1.i] at least one processor configured to:

[1.1] using a cloud provider API, access a block storage volume of a workload maintained in a cloud storage environment;

[1.2] identify an installed software application in the accessed block storage volume;

[1.3] analyze the identified installed software application to determine an associated software version;

[1.4] access a data structure of known software vulnerabilities for a plurality of versions of software applications;

[1.5] perform a lookup of the identified installed software version in the data structure to identify known vulnerabilities; and

[1.6] use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.

2. The system of claim 1, wherein the at least one processor is further configured to implement a remedial action in response to the identified one or more vulnerabilities.

3. The system of claim 2, wherein the remedial action includes transmitting an alert to a device associated with an administrator.

4. The system of claim 1, wherein the at least one processor is further configured to:

query the cloud provider API to determine network accessibility information related to the workload further comprises examining data sources associated with the workload; and

determine the network accessibility information based on the examined data sources.

5. The system of claim 1, wherein to identify the installed software application, the at least one processor is configured to:

extract data from at least one of operating system packages, libraries, or program language libraries; and

identify the installed software application based on the extracted data.

6. The system of claim 1, wherein the at least one processor is further configured to identify a version of the installed software application.

7. The system of claim 1, wherein the identified installed software application includes one or more scripts.

8. The system of claim 1, wherein the data structure includes aggregated vulnerability data.

9. The system of claim 8, wherein the aggregated vulnerability data includes data from one or more third-party vendors.

10. The system of claim 8, wherein the aggregated vulnerability data includes data collected by a scanner.

11. The system of claim 8, wherein the aggregated vulnerability data includes at least one of an advisory, an exploit, a security announcement, or a known bug.

12. A method, comprising:

[12.1] using a cloud provider API, accessing a block storage volume of a workload maintained in a cloud storage environment;

[12.2] identifying an installed software application in the accessed block storage volume;

[12.3] analyzing the identified installed software application to determine an associated software version;

[12.4] accessing a data structure of known software vulnerabilities for a plurality of versions of software applications;

[12.5] performing a lookup of the identified installed software version in the data structure to identify known vulnerabilities; and

[12.6] use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.

13. The method of claim 12, further comprising implementing a remedial action in response to the identified one or more vulnerabilities.

14. The method of claim 13, wherein the remedial action includes transmitting an alert to a device associated with an administrator.

15. The method of claim 12, wherein the method further comprises query the cloud provider API to determine network accessibility information related to the workload, by

examining data sources associated with the workload; and

determining the network accessibility information based on the examined data sources.

16. The method of claim 12, wherein identify the installed software application comprises:

extracting data from at least one of OS packages, libraries, or program language libraries; and

identifying the installed software application based on the extracted data.

17. The method of claim 16, wherein the at least one processor is further configured to identify a version of the installed software application.

18. A non-transitory computer-readable medium storing instructions that, when executed by at least one processor, are configured to cause the at least one processor to perform operations comprising:

[18.1] using a cloud provider API, accessing a block storage volume of a workload maintained in a cloud storage environment;

[18.2] identifying an installed software application in the accessed block storage volume;

[18.3] analyzing the identified installed software application to determine an associated software version;

[18.4] accessing a data structure of known software vulnerabilities for a plurality of versions of software applications; and

[18.5] use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.

I. INTRODUCTION

Petitioner Wiz, Inc. (“Wiz”) respectfully requests review of U.S. Patent No. 11,637,855 (“the ’855 patent”), currently assigned to Orca Security Ltd. (“Orca”). This petition demonstrates claims 1-18 are unpatentable.

The ’855 claims describe well-known techniques for identifying software-specific vulnerabilities present on a workload in a cloud environment and analyzing the risk that the identified vulnerabilities are susceptible to attack from outside the workload. The techniques involve using a cloud provider API to access a block storage volume of a workload to identify software applications that are installed on the workload, analyzing those applications to determine an associated software version, and accessing a data structure of known software vulnerabilities to perform a lookup of the identified installed software version in the data structure to identify known vulnerabilities applicable to the workload. Network accessibility information and at least one port is then used to identify the known and identified vulnerabilities that are susceptible to attack from outside the workload.

This type of software-matching vulnerability identification and subsequent attack risk analysis was already well known prior to the ’855 patent’s priority date, as demonstrated by the combination of Elder and Kim. Elder discloses most aspects of the independent claims, though it does not expressly discuss using cloud provider APIs for its approach or using at least one port to identify vulnerabilities

that are susceptible to outside attack. However, these techniques were well known as shown by Kim, which builds on Elder by referring to it as a background reference. A third reference, Hufsmith, further demonstrates that the techniques described by the '855 patent claims were routine in the art by disclosing the use of additional types of network accessibility information in an analysis assessing the attack risk of identified software vulnerabilities. Furthermore, the dependent claims describe other well-known features, as demonstrated below.

Accordingly, Wiz respectfully requests institution.

II. MANDATORY NOTICES UNDER 37 C.F.R. §42.8

Real Party-in-Interest (37 C.F.R. §42.8(b)(1)): Petitioner Wiz is the real party-in-interest.

Related Matters (37 C.F.R. § 42.8(b)(2)): Wiz is involved in litigation involving Orca patents from a different patent family than those of the '855 patent in *Orca Security Ltd. v. Wiz, Inc.*, No. 1-23-cv-00758 (DDE), filed and served on July 12, 2023. Wiz has also filed the following IPR petitions challenging Orca patents unrelated to the '855 patent: IPR2024-00220, IPR2024-00863, IPR2024-00864, IPR2024-00865, IPR2024-01109, IPR2024-01191, IPR2024-01190.

Lead and Back-Up Counsel (37 C.F.R. §42.8(b)(3)):

Lead Counsel: Matthew A. Argenti (Reg. No. 61,836)

Back-Up Counsel: Michael T. Rosato (Reg. No. 52,182); Wesley E. Derryberry (Reg. No. 71,594); Tasha M. Thomas (Reg. No. 73,207); Joseph M. Baillargeon (Reg. No. 79,685).

Service Information—37 C.F.R. §42.8(b)(4): Wiz consents to electronic service. Please direct all correspondence to lead and back-up counsel at the contact information below. A power of attorney accompanies this petition.

E-mail: margenti@wsgr.com; mrosato@wsgr.com; wderryberry@wsgr.com; tthomas@wsgr.com; jbaillargeon@wsgr.com

Post: WILSON SONSINI GOODRICH & ROSATI, 650 Page Mill Road,
Palo Alto, CA 94304

Tel.: 650-354-4154 Fax: 650-493-6811

III. CERTIFICATIONS

The '855 patent is available for IPR, and Wiz is not barred or estopped from requesting IPR on these grounds.

IV. IDENTIFICATION OF CHALLENGE; STATEMENT OF PRECISE RELIEF REQUESTED

Wiz seeks cancellation of the challenged claims for the reasons stated below, which are supported with exhibits, including the Declaration of Dr. Angelos Stavrou (EX1002). The claims are unpatentable under 35 U.S.C. §311 and AIA §6 based on at least the following grounds:

Ground	Claims	Basis
1	1-6, 8-9, and 11-18	§103(a): obviousness over Elder and Kim.
2	1-18	§103(a): obviousness over Elder, Kim, and Hufsmith.

V. THE '855 PATENT

The '855 patent issued from U.S. Application No. 17/657,972 (“the '972 Application”), filed April 5, 2022. EX1001, Face. The '972 application claims priority to Provisional Application No. 63/180,048, filed April 26, 2021. The '855 patent thus has an effective filing date no earlier than April 26, 2021, and is subject to AIA §102 and §103. *Id.*; EX1002, ¶20.

The '855 patent describes techniques for securing a cloud infrastructure, including “a method of operating a cybersecurity system performing a side scanning function to protect against potential vulnerabilities.” EX1001, 1:15-18, 22:31-33, Fig. 5; EX1002, ¶46. The specification describes well-known software-matching techniques in which software installed on a workload, along with the installed software’s version, is identified and the information is then compared against a list of known software vulnerabilities. EX1001, 22:55-60, 23:56-24:18; EX1002, ¶¶40, 47. The specification also describes the well-known concept of determining the risk that a vulnerability can be exploited by explaining that network accessibility information and at least one port may be used to identify one

or more vulnerabilities “susceptible to attack from outside the workload.” *Id.*, 32:12-26; *see also id.*, 22:60-67 (determining “an avenue for potential vulnerability 513 to access and infiltrate” the workload), 24:19-35, 24:47-62; EX1002, ¶¶42-45, 47.

The ’855 patent includes 18 claims. Claims 1, 12, and 18 are independent. Claims 12 and 18 essentially mirror claim 1, but whereas claim 1 is written as a system claim, independent claim 12 is directed to a method, and independent claim 18 is directed to a computer-readable medium. The dependent claims add other conventional aspects of cybersecurity and cloud computing. EX1002, ¶¶48-49.

A. Prosecution History

During prosecution, the claims of the ’972 application faced rejections under §§102 and 103. The last limitation of the independent claims originally recited that at least one of three actions were performed as part of the claimed cybersecurity analysis: (1) “query the cloud provider API to determine network accessibility information related to the workload”; (2) “identify at least one port on which the vulnerable application is accessible”; or (3) “use network accessibility information and at least one port to identify one or more vulnerabilities susceptible to attack from outside the workload.” EX1004, 770-71, 1234; EX1002, ¶50.

In an Office Action, the Examiner rejected the claims as anticipated by Dye (U.S. Pub. No. 2020/0389482), asserting that Dye disclosed the first of the three

recited actions (i.e., “query the cloud provider API to determine network accessibility information related to the workload”). EX1004, 580-82, 1030-31; EX1002, ¶50. The applicant amended the claims to remove recitation of the first action, but the Examiner responded, in an Advisory Action, that Stopel (U.S. Pub. No. 2019/0116199) taught the second recited action (i.e., “identify at least one port through which an application is accessible”). EX1004, 548, 563 (claim amendment).

The applicant then removed the second recited action from the claims, resulting in only the third action—which recited “network accessibility information” rather than “network accessibility information related to the workload” recited in the first action—remaining in the claims. *Id.*, 461-62. The applicant also amended the third action to further recite “use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.” *Id.* The applicant distinguished Stopel by arguing “Stopel teaches a host device using an at least one port to attempt to access an application to determine whether a vulnerability exists based on an application’s misconfiguration, ... rather than identifying ‘one or more of the known and identified vulnerabilities’ from a ‘data

structure of known software vulnerabilities.’” *Id.*, 468. The Examiner subsequently allowed the claims. *Id.*, 16-25; EX1002, ¶51.

VI. NO BASIS EXISTS FOR DENIAL UNDER 35 U.S.C. §325(d)

Under the two-part *Advanced Bionics* framework, §325(d) analysis considers several factors to determine:

- (1) whether the same or substantially the same art previously was presented to the Office or whether the same or substantially the same arguments previously were presented to the Office; and (2) if either condition of [the] first part of the framework is satisfied, whether the petitioner has demonstrated that the Office erred in a manner material to the patentability of challenged claims.

Advanced Bionics, LLC v. Med-El Elektromedizinische Geräte GmbH, IPR2019-01469, Paper 6 at 8 (precedential); 35 U.S.C. §325(d).

Elder and Kim were not presented to the Office and Hufsmith, while disclosed by the applicant in an IDS, was not discussed by the Examiner. See EX1001, (56); EX1004, 618. The references are also not cumulative of references considered during prosecution. The Office thus did not consider any of the grounds presented herein. The Office also lacked additional evidence discussed herein, including the declaration provided by Wiz’s expert, Dr. Stavrou.

Allowance of the claims also constituted material error under part two of the *Advanced Bionics* test. The Examiner allowed the claims of the ’855 patent

because the applicant distinguished a reference as not disclosing identifying network accessible vulnerabilities that first were identified from a data structure of known software vulnerabilities. *See* EX1004, 468; *supra*, §V.A. However, as discussed in more detail below, Elder teaches first identifying software vulnerabilities present on a workload using a data structure of known vulnerabilities and then assessing each of those vulnerabilities for susceptibility to outside attack by using network accessibility information related to the vulnerability. *Infra*, §XI. Moreover, Kim and Hufsmith describe the well-known practices of also using port status information and workload-specific network accessibility information as part of a security risk analysis. *Infra*, §§XI, XII. The claims therefore should not have issued, and they would not have issued if the Examiner had considered the present grounds.

VII. LEVEL OF ORDINARY SKILL

For purposes of this petition, Wiz assumes a priority date of April 26, 2021. A POSA as of April 2021 would have held at least a bachelor's degree in computer science, computer engineering, electrical engineering, or a related field, and would also have 2-3 years of professional experience working with cyber security analysis and virtualization. Additional experience could compensate for less education and vice versa. Relevant work experience includes, for example, malware analysis, security analysis of cloud computing systems, and security

analysis of VMs. EX1002, ¶¶21-22. Dr. Stavrou meets these requirements and is qualified to credibly opine on the state of the art and the POSA's perspective. *Id.*, ¶¶1-19; *see also id.*, ¶¶2-5 (qualifications); EX1003 (Stavrou CV). Section IX below summarizes the state of the art, including background knowledge that would have informed a POSA's understanding of the references' teachings applied herein.

VIII. CLAIM CONSTRUCTION

Claim terms are given their ordinary and customary meaning, consistent with the specification, as a POSA understood them. 37 CFR §42.100(b); *Phillips v. AWH Corp.*, 415 F.3d 1303, 1312-13 (Fed. Cir. 2005) (en banc). This petition applies the ordinary and customary meaning of the claim terms. *See also* EX1002, ¶52.

IX. BACKGROUND

A. Virtualization and Cloud Computing

Cloud computing and virtualization technology were well known long before 2021. EX1002, ¶¶23-24; EX1008, 1; EX1009, xxiii; EX1010, 2; EX1011, 35. The physical infrastructure for cloud computing was often provided by data centers that included large collections of physical resources. EX1002, ¶29; EX1008, 19; EX1016, 229.

Cloud systems typically used a “virtualization” layer that abstracts the underlying resources to efficiently manage the operation of multiple applications across multiple physical servers. EX1002, ¶¶25-29. Each physical server could

emulate multiple physical computers, called a “virtual machine” (VM), running their own operating system/applications. EX1002, ¶25; EX1009, xxiii; EX1010, 2. Software containers were another form of virtualization known and widely used at the time. EX1002, ¶25; EX1012, ii; EX1013, 1:16-35. Desktop, server, cloud, and datacenter providers routinely used many different virtualization solutions and products. EX1002, ¶25; *see also id.*, ¶¶26-29 (detailing three-layered framework of cloud computing) (citing EX1008, 18-19, 94-95; EX1014, 29; EX1015, 55-58, 62-66, 118, 138, 164-66; EX1016, 229).

Off-the-shelf virtualization tools (e.g., VMWare® VirtualCenter) routinely used APIs to query information about a cloud-based resource, such as a VM. EX1002, ¶30. These tools allowed users to select a particular VM to obtain more detailed information about the VM based on API calls. *Id.*; EX1017, 445-46; EX1018, ¶¶21, 42. Moreover, common cloud/virtualization platforms included APIs that were routinely used for querying information about virtual resources and used as building blocks for more complex tasks, such as performing scans of the virtual disks of a target VM. EX1002, ¶31; EX1019, 13, 23, 32-33, 53-56, 68-69; EX1017, 445-46.

B. Cyber Security

Traditional security systems sought to improve security by identifying security risks including vulnerabilities present on the resource. EX1002, ¶¶32-33,

37. Security scans commonly searched for known risks—e.g., those in published lists such as the Common Vulnerabilities and Exposures (“CVEs”)—and unknown risks that might be indicated, for example, by behavioral abnormalities. EX1002, ¶¶33-34; EX1020, 1-3; EX1021, 1; EX1022, 12; EX1023, 9. Virtual machines were known to be subject to at least the same security risks as nonvirtualized machines. EX1002, ¶¶35-36; EX1024, ES-2, 3-1, 4-1, 4-3; EX1025, 8; EX1026, 171-72; EX1008, 45-46; EX1027, 321.

A POSA would have been familiar with a variety of scanning techniques. EX1002, ¶37; EX1028, 116; EX1029, 22; EX1030, ¶¶27, 43; EX1033, 2; EX1032, ¶43. Both agent-based techniques—in which a security application, or agent, scans the same computer in which it has been installed—and agentless techniques were well-known and commonly used. EX1002, ¶¶38-39; EX1034, 5643; EX1033, 1; EX1032, ¶¶5-6. For example, virtual machine introspection (“VMI”) is an agentless technique that was commonly used in virtualized environments and known to have several advantages (e.g., increasing efficiency and keeping the security software isolated from the potentially compromised guest VM). EX1002, ¶39; EX1033, 1; EX1034, 5643-44; EX1035, 389; EX1036, 3:56-65; EX1024, 3-3; EX1037, 10:9-10; EX1038, 133-34.

Security systems typically evaluated the computer system to identify different types of vulnerabilities, such as checking the configuration files of

operating systems and installed applications to identify out-of-date software that needed to be patched. EX1002, ¶40; EX1039, 3:50-54, 5:28-32; EX1040, 31-36. Matching system data against public repositories of known vulnerabilities was a common way to detect security issues such as applications (or versions of applications) known to be vulnerable or files infected with malware, and various types of matching techniques were well known and routinely used. EX1002, ¶40; EX1041, 2:24-67; EX1042, 2:45-51; EX1059, ¶¶41-51, 56; EX1047, 1-5, 7; EX1043, 14:58-15:18; EX1044, 3:56-4:24, 11:27-47.

Security systems also commonly identified and prioritized risks based on multiple factors, including factors based on the network accessibility of a vulnerability present on an asset. EX1002, ¶41; EX1039, 6:9-18; EX1045, 1:5-42, 4:24-6:44, Figs. 2-4; EX1046, ¶¶48, 75-77. For example, it was known that vulnerabilities often existed in software applications installed on an asset and identifying such vulnerabilities was important due to the risk that the vulnerability could be exploited by an outside attacker, resulting in unauthorized access into the asset. EX1002, ¶42; EX1048, ¶¶3, 37, 44, 53, 60, 79-80; EX1039, 1:18-31, 3:50-57. Thus, well before 2021, POSAs appreciated that identifying ways in which an attacker could exploit an existing software vulnerability over a network allowed administrators to understand the most at-risk vulnerabilities and assets, which

would help guide remediation efforts. EX1002, ¶42; EX1048, ¶¶21, 29; EX1045, 3:28-42.

Information known to be useful in providing insight into the risk that an existing software vulnerability was susceptible to outside attack was the type of network accessibility needed to exploit a known vulnerability. EX1002, ¶43. For example, risk analysis systems often used the Common Vulnerability Scoring System (“CVSS”), which was “an open framework for communicating the characteristics and severity of software vulnerabilities.” EX1049, 3; *see also* EX1002, ¶43; EX1048, ¶55; EX1045, 4:36-50; EX1045, 3:50-54; EX1007, ¶¶116-17; EX1056, 2. Each known software vulnerability (identified by a given CVE identifier) was associated with a CVSS score that included a numerical score and a vector string used to derive the numerical score, with the vector string being comprised of a number of metrics regarding the exploitability characteristics of the associated vulnerability. EX1002, ¶43; EX1049, 3-5, Fig. 1; EX1031, 1; EX1050, 7. For example, the base metrics of a CVSS score provided information as to the network conditions needed to exploit a vulnerability, such as the type of network access required to exploit the vulnerability (e.g., whether an attacker needed local access or could exploit the vulnerability remotely), the types of access conditions needed, and the privileges required to successfully exploit the vulnerability. EX1002, ¶43; EX1049, 6-7, Tables 1-3; EX1050, 7-9; EX1007, ¶117.

Another type of information known to be useful in accessing the risk of a successful attack was the network port settings for the asset having the vulnerability. EX1002, ¶44; EX1048, ¶37; EX1006, ¶15; EX1056, 2. For instance, it was known that software applications commonly communicated through open ports to receive data, but open ports represented a point of entry for an outside attacker to exploit vulnerable applications communicating through those ports. EX1002, ¶44; EX1051, ¶¶5-8; EX1052, ¶6; EX1057, ¶¶39, 57; EX1058, 4:10-36, Fig. 4; EX1048, ¶22. Thus, it was appreciated that the status of a port (e.g., open or closed) on an asset provided information relevant to assessing whether a software vulnerability was susceptible to outside attack. EX1002, ¶44.

It was also commonplace for security systems to use additional environment-specific information in determining whether a vulnerable application could be attacked, such as the location and accessibility in the network environment of the asset containing the vulnerable software. EX1002, ¶45; EX1048, ¶¶15, 22, 55-56; EX1045, 3:43-47, 5:16-6:1. For example, whether the asset has Web-connectivity or the asset's placement relative to other resources in the network were considerations known to provide additional contextual information of the risk that a particular software vulnerability was susceptible to attack. EX1002, ¶45; EX1045, 4:41-46, 8:47-50; EX1048, ¶¶59; EX1039, 5:4-22, 8:46-9:3; EX1007, ¶¶92, 140-41; EX1059, ¶¶76-81.

X. PRIOR ART

A. Elder (U.S. Pub. No. 2014/0189873, EX1005)

Elder was filed May 21, 2010, and published July 3, 2014. It is therefore prior art under 35 U.S.C. §§102(a)(1)-(2).

Elder describes systems and methods for automated risk analysis that identifies and prioritizes vulnerabilities found in host devices located in a network environment, which can be a cloud computing environment. EX1005, Abstract, ¶¶5-6, 27, Fig. 1. As summarized in Kim, Elder “discloses technology related to an automation system which collects the configuration information of a host, analyzes the collected information based on information stored in a vulnerability database (DB), and calculates vulnerability scores.” EX1006, ¶10; EX1002, ¶53.

Elder’s risk analysis system first accesses host configuration information of a host, which includes “configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each.” EX1005, ¶¶7, 28, 33-34, 37-38, Fig. 3. Using the host configuration information, the system then queries a vulnerability database of known vulnerabilities associated with software applications and performs a lookup of the known vulnerabilities that match the software applications and associated versions on the host. *Id.*, ¶¶7, 29-30 (system “maintains an updated database of vulnerabilities” and “correlates the host configuration data against the vulnerability database data”), 32, 39

(vulnerability database reports matched vulnerabilities that include “a plurality of common vulnerability enumerations (CVEs)”), 58-59, Figs. 2, 4; EX1002, ¶54.

After identifying the known vulnerabilities that apply to the host, Elder teaches calculating risk scores “for the host and each software product of the host.” EX1005, ¶¶8, 48; EX1002, ¶55. The risk scores are based on vulnerability scores associated with each identified vulnerability, which can be the base metrics of a CVSS score. EX1005, ¶¶8, 44-49, 62-64, Figs. 2, 6. The risk scores and associated metrics are used to determine, for example, “a software product contributing the most risk to an enterprise, a host contributing the most risk to the enterprise, and a vulnerability contributing the most risk to the enterprise.” *Id.*, ¶8; *see also id.*, ¶¶6, 31, 50-51 (risk quantification determines “the least accessibility needed” of the vulnerabilities present on a host), 54, Fig. 8; EX1002, ¶55.

B. Kim (U.S. Pub. No. 2016/0092679, EX1006)

Kim was filed on July 6, 2015, and published March 31, 2016. It is therefore prior art under §§102(a)(1)-(2).

Kim describes systems and methods for handling vulnerabilities associated with virtual machines in cloud environments. EX1006, Abstract, ¶¶3 (systems and methods “automatically inspect the vulnerability of a virtual machine”), 12, 41; EX1002, ¶56. To identify vulnerabilities on the virtual machine, the system is configured to collect “collection target information ... from the virtual disk and

memory of each virtual machine” using an API. EX1006, Abstract, ¶¶13-15, 23-25, 40, 47-48, 50-51, 53, 66-67, Figs. 2 (virtual disk read/write API), 3 (virtual memory read API), 4. The collection target information may include, for example, “main file information,” “process execution information, kernel memory information, and network port status information of the virtual machine.” *Id.*, ¶¶14-15; *see also id.*, ¶¶22 (collection target information “may include main configuration files of registry and an operating system, an application configuration file, network connection information, and process execution information”), 44, 47, 50; EX1002, ¶57.

Kim teaches that the system inspects the collected information for vulnerabilities according to vulnerability inspection criteria, which can include “software version management status, patch application status, operating system configuration status, network connection status, and process execution status, depending on attributes.” EX1006, ¶¶21; *see also id.*, ¶¶16, 26, 44, 54; EX1002, ¶58.

C. Hufsmith (U.S. Pub. No. 2020/0097662, EX1007)

Hufsmith was filed on September 28, 2018, and published March 26, 2020. It is therefore prior art under §§102(a)(1)-(2).

Hufsmith describes security assessments that detect, prioritize, and report security risks associated with virtual resources (e.g., containers or VMs) that

implement distributed applications. EX1007, ¶¶3-10, 28-37, 149; EX1002, ¶59.

The security assessment may use several scanners (e.g., a static analysis scanner, dynamic analysis scanner, or configuration scanner) to detect multiple types of vulnerabilities, including identifying installed applications that match vulnerable applications listed in public repositories. EX1007, ¶¶82, 84, 121, 160

(embodiments include “performing a lookup on existing vulnerability information in CVE and CWE databases”); EX1002, ¶60.

The security assessment also evaluates and prioritizes each detected vulnerability for overall risk. EX1007, ¶¶6-7, 109; EX1002, ¶61. For example, Hufsmith discloses the system may include a CVE evaluator that assesses CVE properties, which represent “known potential CVEs or other known vulnerabilities and exposures” detected in the asset, and determines “one or more CVE weights” that may be based on metrics describing the CVE, such as a CVSS score vector. EX1007, ¶¶113, 116 (metrics may include “[e]xploitability metrics” that “describe, in part, the thing that is vulnerable and the ease and technical means by which it can be exploited”); *see also id.*, ¶¶117-20 (exploitability metrics can include CVSS base metrics); EX1002, ¶61.

Hufsmith further teaches that the weights associated with the vulnerability can be adjusted “based on contextual properties such as an execution environment” of the asset “or other properties (e.g., historical information) that may implicate an

increase or decrease in risk associated with a given vulnerability.” EX1007, ¶¶92, 94, 135-42. Contextual properties regarding the asset’s execution environment may include, for example, an asset’s network location (e.g., behind a firewall or external-facing), which can “increase[] or decrease[] the likelihood of metrics and values of a potential vulnerability.” *Id.*, ¶¶140-141 (describing, for example, if the asset’s execution environment is “behind a company firewall and for internal use, a weight may be specified for metrics and values implicating a vulnerability to a DDOS attack” that “may discount those metrics and values as a threat”). Contextual properties regarding historical information may include a history of risk scores, which “may indicate trends in the security” of the asset as new vulnerabilities are identified or properties are updated. *Id.*, ¶¶105-06, 139; EX1002, ¶62.

XI. GROUND 1: CLAIMS 1-6, 8-9, AND 11-18 WERE OBVIOUS OVER ELDER AND KIM

The combination of Elder and Kim teaches or suggests all limitations of claims 1-6, 8-9, and 11-18 as a whole. Elder describes systems and methods for automatically assessing the security of hosts on a network environment, such as a cloud computing environment. *Supra*, §X.A; EX1002, ¶63. The security assessment includes accessing host configuration information to identify the software applications installed on the host, determine their associated versions, and compare the information to a database of known software vulnerabilities to identify

vulnerabilities of the host. EX1002, ¶63. Elder further teaches that the identified vulnerabilities are analyzed to quantify their risk by retrieving CVSS score metrics for each vulnerability and using network accessibility information related to the vulnerability provided by the CVSS score metrics. *Id.* Kim builds on Elder, discussing the benefits of using a cloud provider API to access host configuration information and further teaching using network port status information of the host as part of the security assessment. *Supra*, §X.B; *see also infra*, §XI.A; EX1002, ¶63. A POSA had reason to combine Elder and Kim to, for example, perform software-based vulnerability risk assessments, as taught by Elder, while further applying Kim's teachings involving using a cloud provider API and using at least one port as part of the risk assessment. *Infra*, §§XI.A-B.

A. Reasons to Combine Elder and Kim

As an initial matter, the references are analogous art. Elder describes cyber security techniques that identify vulnerabilities in host software configurations and assess their risk to a cloud environment. *Supra*, §X.A; EX1005, Abstract, ¶¶5-8, 24, 28, 30. Kim similarly describes cyber security techniques that identify vulnerabilities based on the configurations of assets in a cloud environment. *Supra*, §X.B; EX1006, Abstract, ¶¶3, 13,-15, 22. Both references are thus in the same field as the '855 patent and are pertinent to issues addressed by the '855

patent. EX1001, 1:15-18 (“Disclosed embodiments include new systems and methods for securing cloud infrastructure”), 22:3-33; EX1002, ¶¶64-65.

Arriving at the claims involved using known elements according to their known functions to achieve predictable results. *KSR Int’l Co. v. Teleflex Inc.*, 550 U.S. 398, 416-17 (2007). *Supra*, §§IX.A-B. Security risk assessments that identified software applications installed on a workload, determined their associated versions, and matched the information to a vulnerability database to identify vulnerabilities of the workload, as taught by Elder, were well-known, predictable techniques. *Supra*, §§IX.B, X.A; EX1002, ¶¶40, 66 (citing EX1048, ¶¶37, 44, 53, 60, 79-80; EX1039, 3:50-57; EX1020, 1-3; EX1041, 2:24-67; EX1042, 2:45-51, 8:1-30; EX1044, 3:56-4:24, 11:27-47; EX1059, ¶¶41-51, 56). In addition, using network accessibility information to determine the susceptibility of an attack from outside the workload, as also taught by Elder, was a known and predictable technique. *Supra*, §§IX.B, X.A; EX1002, ¶¶42-45, 66 (citing EX1049, 3; EX1048, ¶¶15, 22, 55-56, 59; EX1039, 4:36-50, 5:4-22, 8:46-9:3; EX1045, 3:43-54, 4:41-46, 5:16-6:1, 8:47-50; EX1007, ¶¶92, 116-17, 140-41; EX1059, ¶¶76-81). Using a cloud provider API to access configuration information of the workload and using at least one port as part of a security assessment, as taught by Kim, were similarly well-known and predictable techniques. *Supra*, §§IX.A-B, X.B; EX1002, ¶66 (citing EX1048, ¶37; EX1051, ¶¶5-8; EX1052, ¶6; EX1057,

¶¶39, 57; EX1058, 4:10-36; EX1018, ¶21). The combination uses these techniques to predictably perform their known functions since the references describe analogous security assessments, and using Kim’s techniques of collecting host configuration information using a cloud provider API and using at least one port to assess attack risk in security assessments like those of Elder would not meaningfully change their function or result. EX1002, ¶66.

A POSA would also have been motivated to combine Elder and Kim for several reasons. EX1002, ¶67.

First, while an obviousness showing requires no express teaching, suggestion, or motivation to combine references, Kim provides one here. *KSR*, 550 U.S. at 419. Kim identifies Elder as a “related preceding technology” disclosing an “automation system, which collects the configuration information of a host ... and calculates vulnerability scores.” EX1006, ¶10. As explained herein, among Kim’s teachings relevant to improved vulnerability analysis are use of APIs and consideration of ports. Kim’s explicit reference to Elder as a related technology and subsequent disclosure regarding improvements on preceding approaches provide an express teaching, suggestion, or motivation to combine the teachings of the two references, and at minimum satisfies the “expansive and flexible approach” to the obviousness analysis *KSR* requires. *KSR*, 550 U.S. at 415; *see also* EX1002, ¶67.

Moreover, beyond Kim’s explicit reference to Elder, as a technical matter a POSA would have been motivated to use a cloud provider API, such as taught by Kim, to access host configuration information stored in a block storage volume of a workload, such as taught by Elder. EX1002, ¶¶68. Kim explains that systems for performing automated vulnerability inspections of cloud resources, like VMs, were known in the art. EX1006, ¶¶5-11 (inspections including maintenance of the latest security patches of an OS and applications and the configuration of major security may be automated). Such systems often used an agent installed in each VM, but Kim explains that an agent carried with it a risk that “the results of inspection may be forged by a malicious user or malicious code.” *Id.*, ¶9. Elder describes such an agent-based system, using an installed agent to gather host configuration information for performing security assessments. EX1005, ¶¶33 (“[h]ost 202 includes agent 216”), 34 (“agent 216 may collect configuration information”), 77, Fig. 2; EX1006, ¶10 (referencing Elder); EX1002, ¶¶68.

Kim improves upon techniques that rely on agents to collect host configuration information by using an agentless method, which uses APIs provided in the cloud environment to access and gather the host configuration information needed to perform security assessments. EX1006, ¶¶8-9, 12-15. As described above, consistent with Kim, the benefits of agentless scanning methods were already appreciated in the art. *Supra*, §IX.A; EX1002, ¶¶38-39, 69.

In view of the references’ combined teachings, a POSA would have been motivated to use a cloud provider API, such as taught by Kim, to obtain host configuration information for vulnerability assessment, such as taught by Elder. EX1002, ¶70. For example, Elder teaches that host configuration information, such as installed applications, can be obtained from files stored on the host, such as “the registry of a system with the Windows® operating system.” EX1005, ¶34 (collecting “data regarding the operating system, applications that are installed, operating system hotfixes/updates, and application patches”); *see also id.*, (teaching configuration data may also be collected based on portable executable and library files). Kim similarly teaches that configuration information may be collected from the same types of stored files through a cloud provider API. EX1006, ¶¶14-15, 44 (“collection target information may be registry and OS main configuration files, application configuration files, network connection information, process execution information, etc.”). A POSA therefore would have readily recognized that accessing the information needed to perform Elder’s vulnerability assessment could be beneficially achieved with an agentless technique by using a cloud provider API. EX1002, ¶70.

Second, a POSA would have sought to combine the teachings of Elder and Kim to more accurately assess the risk of attack from outside the workload. EX1002, ¶71. Elder teaches that “software applications and operating systems

have become increasingly complex and the number of security vulnerabilities has increased,” making risk analysis of security vulnerabilities increasingly important for enterprise networks. EX1005, ¶4. To identify software applications and operating systems that present risk to the network, Elder describes the well-known technique of matching installed software applications to known vulnerabilities associated with such applications. EX1005, ¶¶7-10, 28-30; *see also* EX1006, ¶10; *supra*, §IX.B. In addition to using known software-matching methods, Elder describes further improving upon the risk assessment by analyzing the vulnerabilities that present the most risk to the enterprise. EX1005, ¶¶7-8, 30, 54. More specifically, Elder teaches determining a CVSS score for each identified software vulnerability, which provides metrics on the type of network accessibility needed to exploit the vulnerability, and using those metrics to assess the vulnerability’s risk of attack. *Id.*, ¶¶8, 43-49; *see also supra*, §IX.B; EX1002, ¶71.

Kim similarly acknowledges that systems that automatically perform security checks for vulnerabilities on cloud computing resources, such as VMs, are desirable. EX1006, ¶¶5-7; EX1002, ¶72. Kim also recognizes that matching installed software applications to a vulnerability database and computing the risk of those vulnerabilities were well-known techniques, even describing Elder as an example system. EX1006, ¶10 (stating Elder collects configuration information,

analyzes the collected information based on data stored in a vulnerability database, and calculates vulnerability scores).

Furthermore, Kim expressly teaches gathering additional network-related information, including information relating to the status of network ports on the host, as part of a security assessment. *Id.*, ¶¶15, 22, 50; EX1002, ¶73. A POSA would have recognized that this type of information would further inform the risk that a vulnerability could be exploited by an attacker outside the workload. EX1002, ¶¶44, 73. Specifically, those skilled in the art understood that network port status (e.g., whether a given port was open or closed) provided relevant information as to whether a vulnerable application was capable of receiving data from outside the host, which would increase the risk that the vulnerable application could be exploited. *Supra*, §IX.B; EX1051, ¶¶5-8 (common to employ data ports to facilitate the receipt of data for applications); EX1052, ¶6 (“If open ports are found, these ports may provide access for possible intrusion, and potentially represent a vulnerability that can be exploited by a malicious hacker.”); EX1058, 4:10-56 (explaining attackers seek out open ports to target vulnerable applications); EX1057, ¶¶39, 57 (mitigating risk of attack by closing port access to vulnerable software); EX1048, ¶22 (same); EX1002, ¶73. A POSA therefore would have sought to use additional network-related information, such as at least one port, as part of the risk assessment, as taught by Kim, in calculating the risk

associated with a software vulnerability, such as taught by Elder, because the information would have better informed the extent to which the vulnerability was susceptible to attack from outside the workload. EX1002, ¶73.

A POSA would have reasonably expected success in combining the references to arrive at the claims. As discussed above, software-matching and vulnerability risk assessments, such as those of Elder, and accessing the needed configuration data using a cloud provider API and using the information as part of the risk assessment, such as taught by Kim, were well known and routinely implemented. EX1002, ¶¶30-31, 40-45, 74 (citing EX1048, ¶¶37, 44, 53, 60, 79-80; EX1041, 2:24-67; EX1042, 2:45-51, 8:1-30; EX1044, 3:56-4:24, 11:27-47; EX1018, ¶21). A POSA would have been familiar with the applied techniques, and the combination would not have presented meaningful technical challenges or produced unexpected results. EX1002, ¶74.

B. Independent Claims

1. Preambles

To the extent the preambles are limiting, Elder, alone or in combination with Kim, teaches them. Claim 1 recites a “cyber security system for a cloud environment, the system comprising.” Claim 12 recites a “method, comprising.” Claim 18 recites a “non-transitory computer-readable medium storing instructions

that, when executed by at least one processor, are configured to cause the at least one processor to perform operations comprising.” EX1002, ¶¶75-76.

Elder describes security methods and systems for automated risk analysis of computer systems. EX1005, Abstract, ¶¶3 (embodiments “generally relate[] to digital computer systems and computer system security”), 6-8, 31, claims 1, 17; EX1002, ¶77. The methods and systems “can be implemented in a variety of architectures and configurations,” such as “a cloud computing environment.” EX1005, ¶24; *see also id.*, ¶27, Fig. 1. Kim similarly teaches cyber security methods and systems for a cloud environment. EX1006, Abstract (systems and methods provide “a secure virtual machine use environment for cloud computing”), ¶¶3, 7, 12, 41, Fig. 1; EX1002, ¶77.

For claim 18, Elder’s system includes a computer-readable medium that is non-transitory and stores instructions to cause a computing device to perform the system’s operations. EX1005, ¶¶9-10, 55, 24-25, 71-72, 80-81, Figs. 8-9, claim 9; EX1002, ¶¶78-79.

2. Element 1.i

Elder teaches this element, which recites “at least one processor configured to” EX1002, ¶80. Specifically, Elder’s system may comprise “a computer system having one or more processors” that are configured to execute instructions to perform the operations associated with the automated risk analysis. EX1005,

¶¶9-10, 24, 55, 71-72, 80-81, Figs. 8-9; EX1002, ¶81. These operations—in the context of the proposed combinations in all grounds—include all operations recited in claim 1. EX1002, ¶¶81-82.

3. Elements 1.1, 12.1, and 18.1

The combination of Elder and Kim teaches these elements, which recite “using a cloud provider API, [access/accessing] a block storage volume of a workload maintained in a cloud storage environment.” EX1002, ¶83.

First, Elder teaches accessing a block storage volume of a workload maintained in a cloud storage environment. EX1002, ¶84. Elder’s system performs risk analysis on hosts 102a-c (i.e., workloads) contained within an operating environment 100, which may be a cloud computing environment. EX1005, ¶¶24, 27 (hosts may be “a variety of computing systems”), Fig. 1; *cf.* EX1001, 16:54-56 (“Workloads, as used herein, may refer to systems, devices, or resources in a network”); EX1002, ¶84 (explaining POSA would have understood that hosts 102a-c would include resources common to a cloud environment, such as virtual machines and containers); *supra*, §IX.A.

The system “collects configuration data from hosts 102a-c for risk analysis,” including “details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each.” EX1005, ¶28; *see also id.*, ¶¶33 (agent 216 “automatically collects and accesses host configuration datastore 220

from host 202”), 35-36, 57 (“host configuration information of a host is accessed), 77, Figs. 3, 7, claims 1, 9, 17; EX1002, ¶85. For example, Elder teaches that host configuration information may be obtained through registry files or other stored files. EX1005, ¶34; cf. EX1001, 18:32-34 (“Accessing block storage volumes, as used herein, may refer to obtaining, examining, or retrieving data.”). Such files were understood to be data files maintained in block storage volumes, such as a virtual disk of a virtual machine. EX1002, ¶85; *see also* EX1005, ¶81 (recognizing applications residing on computer systems “are generally stored on and accessed via a computer readable medium, such as a hard disk drive”); EX1006, ¶¶14, 22, 47 (registry files are accessed on the virtual disk of a VM); EX1053, 11:5-33. Therefore, Elder teaches accessing a block storage volume of a workload maintained in a cloud storage environment. EX1002, ¶85; cf. EX1001, 18:26-36 (block storage volumes refer to “data files maintained ... in cloud-based storage environments”), 39:5-7 (block storage volume may be a virtual disk), 39:10-17.

Elder does not explicitly disclose using a cloud provider API to access the block storage volume, but a POSA would have had reason to use a cloud provider API (rather than a host-installed agent) to obtain the host configuration information in view of Kim, as explained above. *Supra*, §XI.A; EX1002, ¶86. Specifically, Kim describes systems and methods for identifying vulnerabilities present in workloads (e.g., virtual machines) in a cloud environment and, to identify such

vulnerabilities, collecting “collection target information” from each VM using an API provided by the cloud environment. EX1006, Abstract, ¶¶3, 12-14 (using “virtual disk read/write Application Programming Interface (API)”), 15 (using “virtual memory read API” to access virtual memory), 24-25, 40, 47-48, 50-51, Figs. 1-4, claims 1-3, 11-13. Like Elder, Kim teaches the collection target information may include “main configuration files of registry and an operating system, an application configuration file, network connection information, and process execution information.” *Id.*, ¶22; *see also id.*, ¶¶44, 47, 50, 66; EX1002, ¶87. The APIs taught by Kim would have been understood to be cloud provider APIs because it is the provider of the cloud environment that provides access to the types of configuration information taught by Elder and Kim. *Supra*, §IX.A; EX1018, ¶21 (virtual machine manager can include APIs to provide, e.g., disk access and other VM features); EX1002, ¶¶30-31, 87; *cf.* EX1001, 16:49-54 (“A cloud provider API, as used herein, may refer to any application program interface that allows an end user to interact with a cloud provider’s service.”). Thus, a POSA would have used a cloud provider API, as taught by Kim, to access a block storage volume of a workload maintained in a cloud storage environment, as taught by Elder. EX1002, ¶¶87-88.

4. Elements 1.2, 12.2, and 18.2

Elder teaches these elements, which recite “[identify/identifying] an installed software application in the accessed block storage volume.” EX1002, ¶89.

The host configuration information accessed by Elder’s system includes “information about a plurality of software applications and an operating system on the host,” which is used to compare to a vulnerability database to identify known software vulnerabilities present on the host. EX1005, ¶7; *see also id.*, ¶¶28 (data includes “configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each”), 34, 37-39, 57, Figs. 3-4, 7, claims 1-2; EX1002, ¶90; *infra*, §§XI.B.6-7. For instance, Elder teaches “high level data regarding the operating system, applications that are installed, operating system hotfixes/updates, and application patches” can be obtained from registry files or may be collected “based on an enumeration of portable executable (PE) files (e.g., executable (.exe) files and dynamic link library (DLL) files) on the file system.” EX1005, ¶34; EX1002, ¶91. A POSA would have understood that this data obtained from the accessed block storage volume allows the system to identify the software applications installed on the host based on the information. EX1002, ¶¶91-92; EX1054, 4-5, 8 (using Windows registry files to obtain list of installed software applications).

5. Elements 1.3, 12.3, and 18.3

Elder teaches these elements, which recite “[analyze/analyzing] the identified installed software application to determine an associated software version.” EX1002, ¶93.

As explained in the previous section, Elder teaches accessing stored host files to identify installed software applications, which are used to compare to a vulnerability database. *Supra*, §XI.B.4; EX1002, ¶94. These stored files also include information of a software version associated with an installed application. EX1005, ¶¶24 (data collected may include “applications, and associated versions of each”), 34; EX1002, ¶94. When querying the vulnerability database to identify vulnerabilities (*see infra*, §§XI.B.6-7), Elder teaches that its system issues a query that identifies each installed application by “the most precise version number available for each software program,” which may include “a plurality of common platform enumerations (CPEs).” EX1005, ¶¶35, 37-38; EX1002, ¶95.

As Dr. Stavrou explains, the CPE was a well-known and routinely used standard that provided a logical construction in which a system could set values for a defined set of attributes to uniquely identify a specific software application. EX1055, 1 (“Common Platform Enumeration (CPE) is a standardized method of describing and identifying classes of applications, operating systems, and hardware devices present among an enterprise’s computing assets.”), 8-9 (“a well-formed

CPE name (WFN)” can “be used to describe a set of products or to identify an individual product”); EX1002, ¶96. These attributes identified, for example, the class of the product (e.g., applications, operating systems, hardware devices), the product’s vendor, the product’s name, and the product’s version. EX1055, 9, 12, 14; EX1002, ¶97 (describing example CPE). Known vulnerabilities listed in public repositories (e.g., the National Vulnerability Database) were routinely linked to a CPE identifier of the software application affected by the vulnerability, and thus, by enumerating an identified software application’s attributes as a CPE, POSAs understood that the application could be more accurately matched to any vulnerabilities known to be associated with the application. See EX1005, ¶¶38-39 (“CPEs are a list of specific product names that is used to precisely identify a specific version of a product,” which allows the system “to map vulnerabilities to only those exact products and versions to which the vulnerabilities apply”), 45 (explaining “each CVE vulnerability enumerates vulnerable configuration elements with a Common Platform Enumeration (CPE) identifier”); EX1031, 2 (vulnerabilities listed in NVD use CPEs to “communicat[e] the hardware and software platforms that may be affected by a given vulnerability,” which “enables users (and automated tools) to quickly identify vulnerabilities that may affect systems under their purview”); EX1002, ¶98.

Because Elder teaches enumerating the identified installed software applications as CPEs, a POSA would have understood that Elder's system analyzes the identified installed software applications to determine associated software versions. EX1002, ¶99. For example, as explained above (*supra*, §XI.B.4), Elder teaches that software applications installed on a host may be identified by a number of stored files, including registry files, executable files, and library files. EX1005, ¶34; EX1002, ¶99. Each of these files could be analyzed to extract more specific data relating to the installed application corresponding to the file, such as the application's version number, which could then be used to set the value for the version attribute of the application's CPE. See EX1005, ¶37 ("Embodiments of the present invention use the most precise version number available for each software program."); EX1055, 12 (values for version attribute should be "copied directly ... from discoverable data"), 14 (describing set functions for adding attribute-value pairs to CPE); EX1002, ¶99 (describing lookup example of version number for an installed application listed in Windows registry) (citing EX1054, 8-9, 22-27, Figs. 8-9). Thus, Elder teaches analyzing the identified installed software application to determine an associated software version (e.g., extracting version-related data), as recited. EX1002, ¶¶99-100; *cf.*, EX1001, 23:41-60, 25:54-26:11 (software version may be identified "by unique version number or designator, recognized source

code, metadata associated with the installed software application files, or similar software-identifying information”).

6. Elements 1.4, 12.4, and 18.4

Elder teaches these elements, which recite “[access/accessing] a data structure of known software vulnerabilities for a plurality of versions of software applications.” EX1002, ¶101.

To perform risk analysis, “a vulnerability database” (i.e., a data structure) is accessed by the system to identify vulnerabilities associated with the installed software applications. EX1005, ¶¶7, 29 (system “maintains an updated database of vulnerabilities”), 37-39, Figs. 2, 4, 7; EX1002, ¶102. The vulnerability database includes data on known software vulnerabilities, such as vulnerabilities listed in the National Vulnerability Database (NVD). EX1005, ¶¶7, 31-32 (vulnerability database “contains information including security vulnerabilities associated with installed software programs”), 38; *supra*, §IX.B (describing conventional use of public repositories of known software vulnerabilities); EX1002, ¶102. Each of the vulnerabilities contained in the database are associated with a particular version of a given software application (i.e., known software vulnerabilities for a plurality of versions of software applications). EX1005, ¶¶29 (database “catalogs vulnerabilities associated with configuration elements including operating systems, applications, and particular versions of each that are vulnerable”), 38 (system

“map[s] vulnerabilities to only those exact products and versions to which the vulnerabilities apply”), 39 (vulnerabilities in database can include “common vulnerability enumerations (CVEs)”); *supra*, §IX.B (known use of CVEs for vulnerability analysis); EX1002, ¶¶102-03; *cf.* EX1001, 14:1-15 (vulnerability database may include vulnerability data from NVD).

7. Elements 1.5 and 12.5

Elder teaches these elements, which recite “[perform/performing] a lookup of the identified installed software version in the data structure to identify known vulnerabilities.” EX1002, ¶104.

After identifying the software applications installed on the host and their associated versions, Elder’s system stores the data in a program datastore. EX1005, ¶¶35-36, Fig. 4; EX1002, ¶105. A risk analysis module then “issues program name query 420 to vulnerability database 412,” which includes the list of programs stored in the program datastore that identifies the installed program names and associated versions (e.g., by their corresponding CPEs). EX1005, ¶¶37-38; *see also supra*, §XI.B.5. The vulnerability database responds to the risk analysis module “with program name results 422 which includes program names and versions that have vulnerabilities.” EX1005, ¶37. Elder teaches that the risk analysis module, in turn, issues a program name query 424 that lists the programs (and associated versions) identified as having vulnerabilities, and the vulnerability

database “responds with list of vulnerabilities 426 which includes the vulnerabilities based on matching the program names with the vulnerabilities” stored in the database (i.e., performing a lookup of the identified installed software version in the data structure to identify known vulnerabilities). *Id.*, ¶39; *see also id.*, ¶¶38 (“The CPEs are cross-listed with and used by the NVD to map vulnerabilities to only those exact products and versions to which the vulnerabilities apply.”), 58-59, Fig. 7; EX1002, ¶¶106-07.

8. Elements 1.6, 12.6, and 18.5

The combination of Elder and Kim teaches these elements, which recite “use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.”¹ EX1002, ¶108.

¹ While element 18.5 recites identifying “one or more of the known and identified vulnerabilities,” claim 18 does not recite performing a lookup “to identify vulnerabilities” (as recited in claim elements 1.5 and 12.5, discussed above in §XI.B.7). EX1002, ¶109. However, during prosecution of the ’972 application,

First, Elder teaches using network accessibility information that includes data from an external data source to identify one or more of the known and identified vulnerabilities that are susceptible to attack from outside the workload. EX1002, ¶110. After matching the known vulnerabilities stored in the vulnerability database to the specific versions of software applications installed on the host, Elder’s system determines a risk score for each of the host’s software products. EX1005, Abstract, ¶¶8, 46, 54, 62-64, Fig. 7; EX1002, ¶110. To determine a risk score, the system retrieves a vulnerability score for each of the known and identified software vulnerabilities, which Elder teaches can include “Common Vulnerability Scoring System (CVSS) scores.” EX1005, ¶¶7-8, 32, 43-46 (“CVSS scores are assigned to each vulnerability with a Common Vulnerability Enumeration (CVE) identifier.”); EX1002, ¶110.

pending claim 40 (issued as claim 18) originally recited the same lookup element recited in claims 1 and 12. EX1004, 1237-38. While this lookup element did not appear in claim 40 in later claim listings, there is no indication that the element was intentionally removed by the applicant. *Id.*, 774. Therefore, this section treats “the known and identified vulnerabilities” recited in element 18.5 as the identified vulnerabilities discussed above in §XI.B.7. EX1002, ¶109.

A POSA would have understood the CVSS score information taught by Elder provided network accessibility information related to a given vulnerability that constituted data from an external data source. EX1002, ¶111. Elder explains that CVSS was a well-known standard “developed by the U.S. Government, National Institute of Standards and Technology (NIST), MITRE, and others” that “provides a universal open and standardized method for rating IT vulnerabilities” and that “the base metrics of CVSS version 2” may be used for Elder’s risk quantification calculations. EX1005, ¶¶45, 47. Elder explains that the CVSS base metrics include data relating to “three different accessibility characteristics (e.g., access vector, access complexity, and authentication)” for a given vulnerability. *Id.*, ¶47; *supra*, §IX.B. Thus, as a third-party set of metrics, the base metrics of the CVSS would have been understood to be “data from an external data source” as recited by the claims. EX1002, ¶111; EX1031, 2 (CVSS is “a set of metrics for communicating the characteristics and impacts of IT vulnerabilities”).

A POSA would have appreciated that each of these accessibility characteristics of the CVSS base metrics provides “network accessibility information” as recited by the claims because the characteristics conveyed information regarding the network access conditions that were needed for an outside attacker to exploit a given vulnerability present on a host. EX1002, ¶112. For example, POSAs understood that the base metrics of CVSS version 2 “capture

how the vulnerability is accessed and whether or not extra conditions are required to exploit it.” EX1050, 7; EX1002, ¶112. The access vector, for instance, indicated the type of network access needed to exploit a vulnerability (e.g., remote or local access), where the more remote an attacker needed to be to attack the host, the greater the score (i.e., the more likely the vulnerability would be susceptible to attack from outside the workload). EX1050, 7; EX1002, ¶112. The access complexity and authentication metrics also provided network accessibility information that was relevant to a vulnerability’s susceptibility to attack, such as the type of network privileges (e.g., elevated privileges) or the extent of network authentication required to access and exploit the vulnerability. See EX1050, 8 (access complexity measures the complexity of privileges needed to exploit the vulnerability), 8-9 (authentication measures the number of times an attacker must authenticate to a host to exploit the vulnerability); EX1002, ¶113. While Elder expressly references CVSS version 2, a version 3 framework had issued by the relevant timeframe. See EX1049; *supra*, §IX.B; EX1002, ¶114. The CVSS version 3 also used base metrics (e.g., attack vector, attack complexity, and privileges) that largely corresponded to the accessibility metrics of version 2. EX1049, 5-8; EX1007, ¶117; EX1002, ¶114. Thus, whether using version 2, as disclosed by Elder, or the available version 3, a POSA would have understood that the CVSS score metrics used in Elder’s risk analysis provided information about

the type of network access needed to exploit the vulnerability (i.e., network accessibility information). EX1002, ¶115.

Elder further teaches using the network accessibility information provided by the base metric characteristics of the CVSS scores as part of its risk analysis. EX1002, ¶116. Specifically, it was known that, in addition to providing a numerical score, CVSS scores included a vector, which provided “a compressed textual representation that reflects the values used to derive the score,” including the base characteristics described above. EX1050, 1; *see also id.*, 2 (“With CVSS, anyone can see the individual characteristics used to derive a score.”), 5 (“the vector should always be displayed with the vulnerability score”); EX1049, iv, 3 (CVSS “provides a way to capture the principal characteristics of a vulnerability, and produce a numerical score reflecting its severity, as well as a textual representation of that score”); EX1002, ¶¶112-16 (describing textual values used to communicate the base metrics forming the CVSS score), 117 (providing specific example of CVSS base metrics vector representation) (citing EX1050, 20-21; EX1049, 23). As Elder explains, these base metrics are communicated to Elder’s system as part of the CVSS score information and “are used for risk quantification calculations.” EX1005, ¶¶44, 47. For example, as detailed more below, the information is used to calculate risk tuples that provide a measure of risk that the identified vulnerabilities present to the cloud environment and also used to

generate an attack graph that identifies the vulnerabilities having “the least accessibility needed” for exploitation. *Id.*, ¶¶48-51; EX1002, ¶118.

A POSA would have understood that Elder’s use of the network accessibility information provided by the CVSS score base metrics allowed for the identification of one or more of the known and identified vulnerabilities susceptible to attack from outside the workload. EX1002, ¶119. As explained above, the base metrics of a CVSS score provide information on the network access conditions required for exploiting a vulnerability by an outside attacker, thus providing a measure of the susceptibility of attack. *Id.* In addition, Elder teaches that the CVSS score base metrics for each identified vulnerability can be used to calculate risk tuples that include “a composite CVSS score component and an aggregate CVSS score component,” which respectively provide “a measure of severity that reflects that only a single exploited vulnerability is needed by an attacker to achieve the effects of compromise” and “a measure of the number of options that are available to an attacker.” EX1005, ¶¶48-49 (using, e.g., “highest accessibility score” when calculating composite CVSS score); EX1002, ¶119. Thus, each of the composite score and aggregate score identifies one or more of the known and identified vulnerabilities susceptible to attack from outside the workload. EX1002, ¶119. Serving as additional evidence that the CVSS score base metrics enable identification of the vulnerabilities susceptible to attack, Elder

explains that the risk quantification score can be calculated “based on attack graph generation,” where the graph can provide information as to “the least accessibility needed” of all vulnerabilities. EX1005, ¶¶50-51; EX1002, ¶120.

Lastly, while Elder does not expressly disclose also using at least one port to identify one or more of the known and identified vulnerabilities susceptible to outside attack, a POSA would have had reason to use this information in view of Kim. *Supra*, §XI.A; EX1002, ¶121. For example, Kim teaches collecting “collection target information” of the host (e.g., a VM), which can include “network port status information,” as part of a security risk assessment. EX1006, ¶¶15, 50. A POSA understood that network port status information refers to the status of a network port on the host (e.g., open or closed). EX1002, ¶121. As explained above, the status of a port was known to inform whether an attacker could exploit a vulnerability present on an installed application. *See* EX1051, ¶7 (“When a port is open, the application bound to it responds to all traffic directed to the port. This is considered to be the most dangerous state for a network, as traffic from any source is allowed to interact with the application.”); EX1052, ¶6 (open ports “provide access for possible intrusion”); EX1058, 4:10-36 (accessible network ports “allow the attacker to perform targeted attacks on clients ... based on the particular software or operating system that are vulnerable”); EX1002, ¶121; *supra*, §§IX.B, XI.A. Thus, in view of Kim, a POSA had reason to use at

least one port as additional information for analyzing the software vulnerabilities on a workload susceptible to outside attack. EX1002, ¶121. The combined teachings of Elder and Kim therefore teach using network accessibility information (e.g., network accessibility information obtained from CVSS score metrics) and at least one port (e.g., the status of ports on a host) to identify the known and identified vulnerabilities susceptible to attack from outside the workload, as recited. EX1002, ¶¶121-23.

Accordingly, claims 1, 12, and 18 were obvious.

C. Dependent Claims

1. Claims 2-3 and 13-14

Claims 2 and 13 depend from independent claims 1 and 12, respectively, and further recite “[implement/implementing] a remedial action in response to the identified one or more vulnerabilities.” Claims 3 and 14 depend from claims 2 and 13, respectively, and further recite “wherein the remedial action includes transmitting an alert to a device associated with an administrator.” Elder alone, or in combination with Kim, teaches these additional limitations. EX1002, ¶124.

After identifying the software vulnerabilities that apply to a host, Elder teaches that, when a risk score is calculated (e.g., a composite or aggregate score), it is automatically transmitted to an administrator. EX1005, ¶¶6, 8, 32, 46 (risk analysis module “automatically provides detailed risk analysis, quantification, and

prioritization”), 63-68 (further reporting an identification of the software product, host, and vulnerability contributing the most risk), 79, 85, Fig. 7, claim 1; EX1002, ¶125. This reporting mechanism allows the administrator to focus remediation efforts (e.g., patches or upgrades). EX1005, ¶¶49, 54 (system presents a user “with actionable intelligence that can be used to prioritize remediation activities”), 69 (system allows administrator to understand how critical and widespread a vulnerability and the level of importance of remediation efforts). Kim similarly teaches “notifying virtual machine users of the inspection results,” which allows a system operator to use the results for security management. EX1006, ¶¶54-57. Thus, Elder alone, or in combination with Kim, teaches implementing a remedial action by transmitting an alert (e.g., automatic notifications of calculated risk scores and associated risk information) to a device associated with an administrator. EX1002, ¶¶125-26.

Accordingly, claims 2-3 and 13-14 were obvious.

2. Claims 4 and 15

Claim 4 depends from claim 1 and recites “wherein the at least one processor is further configured to: query the cloud provider API to determine network accessibility information related to the workload further comprises examining data sources associated with the workload; and determine the network accessibility information based on the examined data sources.” Claim 15 depends

from claim 12, and similarly recites “wherein the method further comprises querying the cloud provider API to determine the network accessibility information related to the workload, by: examining data sources associated with the workload; and determining the network accessibility information based on the examined data sources.” The combination of Elder and Kim teaches these additional limitations. EX1002, ¶127.

As an initial matter, although claim 4 recites that “query the cloud provider API to determine network accessibility information related to the workload *further comprises* ...,” claim 1 does not include this query limitation. Indeed, as noted above, applicant removed such a limitation from the independent claims during prosecution of the ’855 patent. *Supra*, §V.A. Moreover, claims 4 and 15 also newly refer to “network accessibility information related to the workload,” whereas claims 1 and 12 only recite “network accessibility information.” *See id.* (applicant deleting reference to network accessibility information “related to the workload” from the independent claims). Thus, while claims 4 and 15 require the system query the cloud provider API to determine network accessibility information related to the workload, the claims do not require that the system *use* the “network accessibility information related to the workload” to identify the known vulnerabilities susceptible to attack as recited in claim 1. As detailed more below, Kim teaches querying a cloud provider API to determine “network

accessibility information related to the workload.” And, while not required by the claims, a POSA would have further found it obvious from the combined teachings of Elder and Kim to use network accessibility information “related to the workload” to identify the known vulnerabilities that are susceptible to attack.² EX1002, ¶127.

Kim teaches that, in addition to network port status information, the cloud provider API can be queried for other network-related information, including information regarding the network accessibility of the workload (e.g., virtual machine). EX1002, ¶128. For example, Kim teaches that the cloud provider API can be queried for “network connection information” of the workload as part of the collection target information, which a POSA would have understood would include the status of the workload’s network connection. EX1006, ¶¶21 (vulnerability assessment includes “network connection status”),²²; EX1002, ¶128. Kim teaches that the network connection information is determined by examining data from the virtual disk or memory of the VM (i.e., data sources associated with the workload). EX1006, ¶¶14-15.

² Further, Ground 2 explains how Hufsmith also taught a POSA to use network accessibility information related to the workload to determine the risk of outside attack of known vulnerabilities. *Infra*, §XII.

Moreover, as explained above, a POSA had reason to use additional network-related information, such as taught by Kim, as part of a risk analysis assessment, as taught by Elder. *Supra*, §§XI.A, XI.B.8; EX1002, ¶129. A POSA would have understood that information regarding the network connection of the workload (e.g., whether the workload was connected to the Internet) would provide additional information as to whether a vulnerability could be successfully exploited by an outside attacker. See EX1045, 3:63-67, 4:41-46, 5:47-48 (assessing whether workload is “in an Intranet, the Internet, or a local area network” as part of risk assessment), 8:47-50 (a vulnerability “that relies on access via the Web, and an asset that has Web-connectivity” indicates higher risk of attack); EX1002, ¶¶129-30.

Accordingly, claims 4 and 15 were obvious.

3. Claims 5 and 16

Claims 5 and 16 depend from claims 1 and 12, respectively, and further recite, to “identify the installed software application,” the at least one processor is configured to/the method further comprises “[extract/extracting] data from at least one of [operating system/OS] packages, libraries, or program language libraries; and [identify/identifying] the installed software application based on the extracted data.” Elder teaches these additional limitations. EX1002, ¶131.

Elder explains that the host configuration information, from which installed software applications are identified, can be obtained from “dynamic link library (DLL) files” (i.e., libraries). EX1005, ¶34; EX1002, ¶132; *supra*, §XI.B.4. As explained above, Elder also describes enumerating the installed software applications as CPEs for vulnerability identification. EX1005, ¶¶37-38; *supra*, §XI.B.5. To generate the CPEs, a POSA would have understood that the system’s processor would be configured to extract data from the library files and analyze the extracted data to identify the software applications for formatting into a CPE. See EX1018, ¶32 (describing parsing data in retrieved VM images to determine VM attributes for vulnerability scanning); EX1002, ¶¶132-33.

Accordingly, claims 5 and 16 were obvious.

4. Claims 6 and 17

Claims 6 and 17 depend from claims 1 and 16, respectively, and further recite “wherein the at least one processor is further configured to identify a version of the installed software application.”³ Elder teaches these additional limitations. EX1002, ¶134.

³ Claim 17 recites “the at least one processor,” but claim 17 indirectly depends from claim 12, which is a method claim that does not recite a processor.

As explained above, Elder’s system is configured to identify a version of the installed software applications identified from the host configuration information so that the applications can be matched to known vulnerabilities stored in a vulnerability database. EX1005, ¶¶34, 37-39; *supra*, §XI.B.5; EX1002, ¶136. Elder’s system also includes a processor configured to execute the steps recited in claim 1 and its dependent claims, which would perform these steps as part of the method of claim 17. EX1002, ¶¶136-37; *supra*, §XI.B.2.

Accordingly, claims 6 and 17 were obvious.

5. Claim 8

Claim 8 depends from claim 1 and further recites “wherein the data structure includes aggregated vulnerability data.” As explained above, Elder’s system “maintains an updated database of vulnerabilities,” which “catalogs vulnerabilities associated with configuration elements including operating systems, applications, and particular versions of each that are vulnerable” (i.e., aggregated vulnerability data). EX1005, ¶29; *supra*, §XI.B.6; EX1002, ¶¶138-40.

Accordingly, claim 8 was obvious.

Nevertheless, Elder teaches a method in which a version of the installed software application is identified. EX1002, ¶135.

6. Claim 9

Claim 9 depends from claim 8 and further recites “wherein the aggregated vulnerability data includes data from one or more third-party vendors.” Elder teaches the vulnerability database may comprise data “provided by the National Vulnerability Database (NVD),” which is a third-party vendor. EX1005, ¶29; EX1002, ¶¶141-43 (citing EX1031, 1).

Accordingly, claim 9 was obvious.

7. Claim 11

Claim 11 depends from claim 8 and recites “wherein the aggregated vulnerability data includes at least one of an advisory, an exploit, a security announcement, or a known bug.” As explained above, Elder’s vulnerability database maintains aggregated vulnerability data, such as data provided by the NVD. *Supra*, §§XI.B.6, XI.C.5-6. The vulnerabilities listed by the NVD includes vulnerabilities that can be exploited by a malicious attacker (i.e., an exploit). See EX1005, ¶48 (risk analysis measures severity of exploited vulnerability), claim 1; EX1002, ¶¶144-45. The NVD was also known to provide advisories and other security announcements. EX1031, 3 (NVD provides resources such as vulnerability advisories, checklists, alerts, vulnerability summaries, and various announcements lists); EX1002, ¶¶145-46.

Accordingly, claim 11 was obvious.

XII. GROUND 2: CLAIMS 1-18 WERE OBVIOUS OVER ELDER, KIM, AND HUFSMITH

This ground builds upon Ground 1 and further applies Hufsmith's teachings of using contextual properties relating to the workload's execution environment, in addition to accessibility information provided by CVSS metrics, to identify vulnerabilities susceptible to attack outside of the workload and using stored aggregated vulnerability data collected by a scanner as part of the risk analysis. The combination does not impact the previous obviousness arguments for limitations beyond those discussed in §§XII.A-B. EX1002, ¶147.

A. Reasons to Combine Elder, Kim, and Hufsmith

The references are analogous art. Hufsmith describes systems and methods for detecting and managing potential vulnerabilities in distributed applications. EX1007, Abstract, ¶¶3-5, 8 (performing CVE analysis), 33 (same), 149 (systems and methods assess risk for cloud-based resources, such as VMs and containers); *supra*, §§XI.A-C (summary of references), XII.A (reasons to combine Elder and Kim). The references are thus in the same field as the '855 patent and reasonably pertinent to issues addressed by the '855 patent. EX1001, 1:15-18, 22:3-33; EX1002, ¶¶148-49.

Arriving at the claimed invention involved using known elements according to their known functions to achieve predictable results. *KSR*, 550 U.S. 398, 416-17 (2007). In addition to the known elements of Elder and Kim discussed above

(*supra*, §§XI.A-B), assessing risk based on environment-specific network accessibility properties relating to a workload and using vulnerability data collected by a scanner, as taught by Hufsmith, was well-known and predictable. *See, e.g.*, EX1045, 3:43-67, 5:6-6:1 (using “environment specific details” in risk analysis to provide “more accurate and meaningful” prioritization than using “just the CVSS score”); EX1048, ¶¶96-97 (asset criticality based in part on asset’s IP address and/or geographic location); EX1039, 6:9-7:10 (prioritizing based on factors such as connection to other assets in the network); EX1046, ¶¶48, 75-77; EX1059, ¶¶76-81; EX1053, 4:14-18 (scanning system can perform “historical analysis of multiple sequential snapshots stored in a database, to identify changes in the virtual computing environment”); EX1002, ¶¶45, 150. Thus, the combination of Elder, Kim, and Hufsmith would have merely involved using familiar techniques (i.e., using environment-specific network properties of a workload and scanner-collected vulnerability data for analysis of the risk of attack) according to their predictable results. EX1002, ¶150.

A POSA also would have been motivated to apply Hufsmith’s teachings to the combination of Elder and Kim to further enhance the identification of the known and identified software vulnerabilities susceptible to attack from outside the workload. EX1002, ¶151.

First, like Elder, Hufsmith teaches scanning assets in a cloud environment for software vulnerabilities (e.g., by identifying applicable CVEs) and calculating a risk associated with the identified vulnerabilities by determining a CVSS score for each vulnerability. EX1007, ¶¶8, 61, 67, 82, 84 (scanners may scan resources for vulnerabilities “identified in public repositories, such as repositories of CVE or CWE vulnerabilities”), 105; EX1002, ¶152. As with Elder, Hufsmith teaches using the basic metrics of the CVSS as part of the risk assessment and thus identifies vulnerabilities susceptible to attack based on network accessibility information relating to the particular vulnerability. EX1007, ¶¶113-20 (CVE evaluator evaluates CVE properties of assets to assign one or more CVE weights, which can be based on CVSS vectors, including metrics “indicating whether an attacker requires (N) network access (e.g., OSI layer 3), (A) adjacent network access (e.g., shared physical or local IP subnet), (L) local network access (e.g., read/write/execute capabilities), (P) physical access (e.g., via peripheral or other physical interaction), or other means”); EX1002, ¶152.

Hufsmith also explains that, in addition to using the network accessibility information provided by CVSS scores for identified vulnerabilities, the risk analysis can be further refined by additionally considering context-specific network properties of the host that contains the vulnerability, such as the host’s execution environment. EX1007, Abstract, ¶¶8, 92 (results engine “may adjust

weights associated with one or more vulnerabilities based on contextual properties such as an execution environment” of the asset), 106. Hufsmith teaches that the context-specific properties can include network accessibility information related to the asset, including, for example, the asset’s relative location and exposure within the network (e.g., whether the asset is behind a firewall or externally exposed). *Id.*, ¶¶92 (“In a specific example, a given vulnerability or a given type of vulnerability may pose an increase in risk over a baseline value if (e.g., if and only if) container functions are exposed externally ...”), 135, 140 (“execution environments 129 may indicate whether the container or distributed application is on an in-band network, out-of-band network, behind a firewall, behind an API, executed on a cloud service or within a company data center, and so on”); EX1002, ¶153.

A POSA would have recognized that using environment-specific network accessibility information, such as taught by Hufsmith, would aid in improving the accuracy of assessing the risk that an outside attacker could successfully exploit a software vulnerability present on a host. EX1002, ¶154. As an example, Hufsmith describes a scenario where “a container image may include a subroutine that [is] registered with a networks socket, but a composition file or other code may indicate that the subroutine is never invoked,” thus providing an indication of a decreased risk that a vulnerability could be exploited due to the subroutine’s dormancy. EX1007, ¶140. As another example, Hufsmith teaches that “a

vulnerability to a DDOS attack” can be less susceptible to attack outside of the workload if “the execution environment of a container i[s] behind a company firewall and for internal use.” *Id.*, ¶141. Thus, a POSA would have recognized the benefits of improving upon Elder’s risk assessment by additionally using network accessibility information relating to environment-specific properties of the workload because doing so would aid in more accurately identifying the known and identified vulnerabilities susceptible to attack from outside the workload. EX1002, ¶154.

Second, in view of Hufsmith’s teachings, a POSA would have also had reason to use other contextual properties such as historical information for risk analysis. EX1007, ¶92 (results engine “may adjust weights associated with one or more vulnerabilities based on ... historical information” of the asset); EX1002, ¶155. For example, Hufsmith teaches that the contextual properties for a given asset may also include a “score history,” such as “one or more historical scores” calculated based on the results of previous risk assessments. EX1007, ¶¶106, 135, 139. Hufsmith explains that, by using historical score data, the risk assessment may identify “trends in the security of the container as new CVEs or CWEs or other exploits of the container are identified or their respective properties updated to either reflect an increase or decrease in vulnerability.” *Id.*, ¶139. The benefit of tracking historical trends of risk scores is also recognized by Elder. EX1005, ¶¶49

(using an “aggregate CVSS calculation,” which “offers defenders a mechanism for measuring progress with respect to each remediation action (e.g., fix install or software upgrade)”), 69 (recalculating risk scores “[w]hen a major new vulnerability is announced” so administrators can “understand how critical and widespread this vulnerability is in his or her enterprise and the level of important[ce] of remediation activities”). A POSA therefore would have had reason to track historical scan data for an asset, as taught by Hufsmith, as part of a risk analysis such as taught by Elder because doing so would allow for greater insight into the overall risk trend for an asset and the impact of remediation efforts. EX1002, ¶155.

A POSA would have had a reasonable expectation of success. EX1002, ¶156. Using environment-specific network accessibility information related to a workload to assess the risk of attack was well understood and routinely practiced. EX1002, ¶156 (citing EX1045, 3:43-67, 5:6-6:1; EX1048, ¶¶96-97; EX1039, 6:9-7:10; EX1046, ¶¶48, 75-77; EX1059, ¶¶76-81). Similarly, as confirmed by Elder, the use of historical scan data to analyze trends in an asset’s risk was also a well-understood and routinely practiced technique. EX1005, ¶¶49, 69; EX1002, ¶156 (citing EX1053, 4:14-18). Therefore, combining the references’ teachings as discussed herein would not have presented meaningful technical challenges or produced unexpected results. EX1002, ¶156.

B. Claims 1, 12, and 18

Elements 1.6, 12.6, and 18.5 recite “use network accessibility information and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, wherein the network accessibility information includes at least one of: data from an external data source, cloud provider information, or at least one network capture log.” The combination of Elder, Kim, and Hufsmith teaches these elements. EX1002, ¶157.

Ground 1 explained how the combination of Elder and Kim teaches using network accessibility information that includes data from an external data source (e.g., CVSS scores and related metrics) and at least one port to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload. *Supra*, §§XI.A, XI.B.8; EX1002, ¶158.

Like Elder and Kim, Hufsmith also teaches scanning a cloud-based asset (e.g., a VM or container) to identify vulnerabilities associated with installed applications and using CVSS score metrics as part of analyzing the risk that the vulnerability presents to the asset. *See* EX1007, ¶¶8, 61, 67, 82, 84, 105, 113-120, 149; EX1002, ¶159. As explained above, Hufsmith teaches that this risk assessment can also consider environment-specific network accessibility information related to the workload (e.g., whether the asset is externally exposed, behind a firewall, within an internal network) to provide a more accurate

assessment of risk. EX1007, ¶¶92, 106, 135, 140-41; *supra*, §§XII.A. A POSA would have understood that the environment-specific network accessibility information taught by Hufsmith was “network accessibility information” related to a workload that includes “data from an external data source” (e.g., data relating to the asset’s operating environment) and “cloud provider information” (e.g., information relating to network access to the workload). *Cf.* EX1001, 25:37-39 (“an external data source may include data from the operating environment of the cloud-based environment”), 25:41-44 (“Cloud provider information may further include information that may identify the network accessibility information 511 vertically or horizontally to fully describe the associated workload.”); EX1002, ¶159. As discussed above, a POSA would have applied Hufsmith’s teachings with the combination of Elder and Kim such that the risk analysis would additionally use environment-specific network properties relating to the workload to identify the known and identified software vulnerabilities susceptible to attack from outside the workload. EX1002, ¶¶160-61.

Accordingly, claims 1, 12, and 18 were obvious for these additional reasons.

C. Claims 4 and 5

Claim 4 depends from claim 1 and recites “wherein the at least one processor is further configured to: query the cloud provider API to determine network accessibility information related to the workload further comprises

examining data sources associated with the workload; and determine the network accessibility information based on the examined data sources.” Claim 15 depends from claim 12 and similarly recites “wherein the method further comprises querying the cloud provider API to determine the network accessibility information related to the workload, by: examining data sources associated with the workload; and determining the network accessibility information based on the examined data sources.” The combination of Elder, Kim, and Hufsmith teaches these additional limitations. EX1002, ¶162.

As explained above, Kim teaches querying a cloud provider API to determine network accessibility information related to the workload (e.g., the workload’s network connection) by examining data sources associated with the workload (e.g., data contained within the workload’s virtual disk and memory). *Supra*, §XI.C.2; EX1002, ¶163. Hufsmith further teaches determining network accessibility information related to the workload (e.g., environment-specific network accessibility information, discussed above, *supra*, §XII.B) by examining data sources related to the workload. EX1002, ¶164. For example, Hufsmith teaches the properties relating to the workload’s execution environment may be determined by examining configuration files of the workload. *See* EX1007, ¶92 (“[T]he results engine 54 may adjust a weight associated with a detected potential vulnerability or type of detected potential vulnerability based on one or more

context properties indicating an intended (or selected or inferred, e.g., based on a Docker compose file or other orchestration configuration) execution environment for the container image.”). A POSA would have queried the cloud provider API, as taught by Kim, to examine these data sources because this type of information was known to be accessible through APIs provided by the cloud environment. *See* EX1006, ¶22 (information obtained through APIs can include configuration files of VMs); EX1002, ¶¶164-65.

Accordingly, claims 4 and 15 were obvious for these additional reasons.

D. Claim 7

Claim 7 depends from claim 1 and further recites “wherein the identified installed software application includes one or more scripts.” Hufsmith teaches this additional limitation. EX1002, ¶166.

Hufsmith teaches the common understanding that applications installed on workloads found in a cloud environment routinely included one or more scripts. EX1002, ¶167. For example, Hufsmith explains that, as part of the risk assessment, configuration files of installed applications may be scanned. EX1007, ¶¶58, 61, 73, 86, 108; EX1002, ¶167. These configuration files may include scripts, as evidenced by Hufsmith’s teaching that the scanners used to scan the configuration files may be selected “for various source code of interpreted languages (e.g., Python™, JavaScript™, and the like).” EX1007, ¶73; *see also id.*,

¶¶59 (describing “execut[ing] scripts that configure applications, for example based on composition files”), 63 (“container images may be specified by a text file, such as an executable text file encoding a script with a plurality of lines”), 69 (“Dockerfile or other script from which a container image is composable”). JavaScript and Python were well-known and routinely used scripting languages. *See* EX1001, 26:6-11 (“These scripts may be processed through various computer readable languages to include Javascript, C#, C++ and other forms of computer code.”); EX1002, ¶167. Thus, as evidenced by Hufsmith, a POSA would have understood that many applications installed in cloud-based resources would have included scripts since scripts were commonly used programming instructions for executing the functions of an application. *See* EX1007, ¶217 (“Program instructions may include a computer program (which in certain forms is known as a program, software, software application, script, or code).... A program may be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document)....”); EX1002, ¶¶167-68.

Accordingly, claim 7 was obvious.

E. Claim 10

Claim 10 depends from claim 8 and further recites “wherein the aggregated vulnerability data includes data collected by a scanner.” The combination of Elder, Kim, and Hufsmith teaches this additional limitation. EX1002, ¶169.

Ground 1 explained how Elder teaches that its system maintains a vulnerability database storing data on known software vulnerabilities (i.e., a data structure that includes aggregated vulnerability data). *Supra*, §§XII.C.5-7; EX1002, ¶170. Hufsmith further teaches maintaining a score database for a scanned workload (e.g., a container). EX1007, ¶¶100-02, 105, 205, Fig. 1B; EX1002, ¶170. The score database stores “CVE properties 111,” which “correspond to a set of identified CVE properties determined by a static analysis scanner or a dynamic scanner.” EX1007, ¶105. The score database also stores a “score history 128,” which “may include one or more historical scores” that “indicate trends in the security of the container as new CVEs or CWEs or other exploits of the container are identified or their respective properties updated to either reflect an increase or decrease in vulnerability.” *Id.*, ¶¶106, 139. Hufsmith thus teaches maintaining, in a database, the CVE properties identified by a scanner for a workload and historical risk scores calculated from vulnerabilities identified from previous scans of the workload. EX1002, ¶170. These types of data represent “aggregated vulnerability data” (e.g., CVE properties and score history) that “includes data collected by a scanner” (e.g., the set of identified CVE properties of the workload determined by a scanner and the historical risk scores derived from vulnerabilities identified in previous scans). EX1002, ¶170.

As discussed above, a POSA would have applied Hufsmith’s teachings to the combination of Elder and Kim such that the risk analysis would account for historical information in scan data relating to the workload in order to analyze trends in the risk scores associated with the workload and the effects of remediation efforts. *Supra*, §XII.A; EX1002, ¶171. Moreover, a POSA would have readily appreciated that a database containing historical scan data, such as taught by Hufsmith, could be included in the same data structure as the vulnerability database taught by Elder. EX1005, ¶¶29, 40 (teaching fix database and vulnerability database may be “part of the same database or data source”). The combination of Elder, Kim, and Hufsmith thus teaches a data structure having aggregated vulnerability data that includes data collected by a scanner. EX1002, ¶¶171-72.

Accordingly, claim 10 was obvious.

XIII. CONCLUSION

The challenged claims are unpatentable. Wiz respectfully requests that IPR be instituted.

Dated: November 1, 2024

Respectfully submitted,

/Matthew A. Argenti/

Matthew A. Argenti, Lead Counsel
Reg. No. 61,836

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

Under the provisions of 37 C.F.R. §42.24(d), the undersigned hereby certifies that the word count for the foregoing Petition for *Inter Partes* Review totals 13,519 which is less than the 14,000 allowed under 37 C.F.R. 42.24(a)(i). In accordance with 37 C.F.R. 42.24(a), this word count does not include any table of contents, table of authorities, mandatory notices under §42.8, certificate of service or word count, or appendix of exhibits or claim listing.

Dated: November 1, 2024

Respectfully submitted,

/Matthew A. Argenti/

Matthew A. Argenti, Lead Counsel
Reg. No. 61,836

PAYMENT OF FEES UNDER 37 C.F.R. §§42.15(A) AND 42.103

The required fees are submitted herewith. If any additional fees are due at any time during this proceeding, the Office is authorized to charge such fees to Deposit Account No. 23-2415.

LIST OF EXHIBITS

Exhibit No.	Description
1001	U.S. Patent No. 11,637,855 to Shua
1002	Declaration of Dr. Angelos Stavrou
1003	<i>Curriculum Vitae</i> of Dr. Angelos Stavrou
1004	File History of U.S. Patent Application No. 17/657,972 ('855 Pat.)
1005	U.S. Publication No. 2014/0189873 to Elder
1006	U.S. Publication No. 2016/0092679 to Kim
1007	U.S. Publication No. 2020/0097662 to Hufsmith
1008	“NIST Cloud Computing Standards Roadmap,” NIST Special Publication 500-291, Version 2 (July 2013)
1009	Wolf, “Virtualization: From the Desktop to the Enterprise” APRESS (2005)
1010	Waldspurger, “Memory Resource Management in VMware ESX Server,” USENIX, Proceedings of the 5th Symposium on Operating Systems Design and Implementation (Dec. 9-11, 2002)
1011	Goldberg, “Survey of Virtual Machine Research,” COMPUTER, Vol. 7, No. 6 (June 1974)
1012	Souppaya, “Application Container Security Guide,” NIST Special Publication 800-190 (Sept. 2017) https://doi.org/10.6028/NIST.SP.800-190
1013	U.S. Patent No. 11,106,784 B2 to Rosendahl
1014	Roberts, “Public Cloud Service Definition,” VMware vCloud® Architecture Toolkit™ for Service Providers, VMware, Inc. (2018)
1015	vSphere Web Services SDK Programming Guide, VMWare, Inc. (April 17, 2018)

Exhibit No.	Description
1016	Wood, “Black-box and Gray-box Strategies for Virtual Machine Migration” NSDI ’07: 4th USENIX Symposium on Networked Systems Design & Implementation, USENIX
1017	VMware VirtualCenter User’s Manual version 1.4, VMware, Inc. (July 7, 2006)
1018	U.S. Publication No. 2013/0247133 A1 to Price
1019	Virtual Disk Programming Guide: vSphere Storage APIs for Data Protection (VADP) 6.0 & Virtual Disk Development Kit (VDDK) 6.0, VMware, Inc. (March 25, 2016)
1020	ITL Bulletin, “Security Patches and the CVE Vulnerability Naming Scheme: Tools to Address Computer System Vulnerabilities,” NIST (Oct. 2002)
1021	ITL Bulletin, “Preventing and Handling Malware Incidents: How to Protect Information Technology Systems from Malicious Code and Software,” NIST (Dec. 2005)
1022	Cichonski, “Computer Security Incident Handling Guide: Recommendations of the National Institute of Standards and Technology,” NIST Special Publication 800-61, Rev. 2 (Aug. 2012) http://dx.doi.org/10.6028/NIST.SP.800-61r2
1023	ENISA Threat Landscape Report 2018, “15 top Cyberthreats and Trends” V1.0 (Jan. 2019)
1024	Scarfone, “Guide to Security for Full Virtualization Technologies: Recommendations of the National Inst. of Standards and Technology,” NIST Special Publication 800-125 (Jan. 2011)
1025	Verizon, “2019 Data Breach Investigations Report” (May 2019)
1026	Di Pietro, “Virtualization Technologies and Cloud Security: Advantages, Issues, and Perspectives,” Springer Nature Switzerland AG (2018) https://doi.org/10.1007/978-3-030-04834-1_9

Exhibit No.	Description
1027	Singh, “Overview of Attacks on Cloud Computing, Int’l. J. of Eng. And Innovative Technology (IJEIT), Vol. 1, Issue 4 (Apr. 2012)
1028	Bairwa, “Vulnerability Scanners: A Proactive Approach to Assess Web Application Security,” Int’l. J. on Computational Sciences & Applications (IJSCA), Vol. 4, No. 1 (Feb. 2014)
1029	Stone, “IT Asset Management,” NIST Special Publication 1800-5 (Sept. 2018) http://doi.org/10.6028/NIST.SP.1800-5
1030	EP Patent No. 3,360,071 B1 to Li
1031	ITL Bulletin, “The National Vulnerability Database (NVD): Overview,” NIST (Dec. 2013)
1032	U.S. Publication No. 2017 / 0372070 A1 to Burdett
1033	Garfinkel, “A Virtual Machine Introspection Based Architecture for Intrusion Detection,” Computer Science Dept., Stanford University (2003)
1034	Tang, “VirtAV: An Agentless Runtime Antivirus System for Virtual Machines,” KSII Transactions on the Internet and Information Systems, Vol. 11, No. 11 (Nov. 2017) https://doi.org/10.3837/tiis.20171.026
1035	Lengyel, “Scalability, Fidelity and Stealth in the DRAKVUF Dynamic Malware Analysis System (2014) https://doi.org/10.1145/2664243.2664252
1036	U.S. Patent No. 8,613,080 B2 to Wysopal
1037	Bauman, “A Survey on Hypervisor-Based Monitoring: Approaches, Applications, and Evolutions,” ACM Computing Surveys, Vol. 48, No. 1, Article 10 (Aug. 2015)
1038	Suneja, “Exploring VM Introspection: Techniques and Trade-offs,” ACM, VEE’15 (Mar. 2015) http://dx.doi.org/10.1145/2731186.2731196

Exhibit No.	Description
1039	U.S. Patent No. 9,141,805 B2 to Giakouminakis
1040	Carroll, “Challenges in Modern Digital Investigative Analysis,” published in Forensic Science and Forensic Evidence I, U.S. Att’ys Bull. Vol. 65 No. 1 (January 2017)
1041	U.S. Patent No. 10,055,576 B2 to Milner
1042	U.S. Patent No. 9,824,216 B1 to Khalid
1043	U.S. Patent No. 8,984,478 B2 to Epstein
1044	U.S. Patent No. 9,749,349 B2 to Czarny
1045	U.S. Patent No. 9,692,778 B1 to Mohanty
1046	U.S. Publication No. 2017/004826 A1 to Hovor
1047	Bourquin, “Binslayer: Accurate Comparison of Binary Executables,” 2nd ACM SIGPLAN Program Protection and Reverse Engineering Workshop (2013)
1048	U.S. Publication No. 2013/0191919 A1, published July 25, 2013, to Basavapatna
1049	International Telecommunication Union, “Common vulnerability scoring system 3.0,” Recommendation ITU-T X.1521 (Mar. 2016) https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1521-201603-I!!PDF-E&type=items
1050	Mell P., Scarfone K. and Romanosky S., “A Complete Guide to the Common Vulnerability Scoring System Version 2.0” (July 2007) https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198
1051	U.S. Publication No. 2003/0154269 A1, published August 14, 2003, to Nychama
1052	U.S. Publication No. 2004/0015728 A1, published January 22, 2004, to Cole

Exhibit No.	Description
1053	U.S. Patent No. 11,216,563 B1 to Veselov
1054	Risto, J., “Windows Installed Software Inventory: Gathering the Information Needed for the 20 Critical Controls,” Global Information Assurance Certification Paper (August 2016)
1055	Cheike B.A., Waltermire D. and Scarfone K., “Common Platform Enumerations: Naming Specification Version 2.3,” NIST (August 2011)
1056	Archived copy of “Vulnerability Management and Scanning: Monitor networks to keep vulnerabilities in check,” published by RAPID7 and archived on July 28, 2018, by the Internet Archive’s “Wayback Machine”
1057	U.S. Publication No. 2020/0228560, published July 16, 2020, to Murthy
1058	U.S. Patent No. 9,686,296 B1 to Murchison
1059	U.S. Publication No. 2014/0245376 A1 to Hibbert

CERTIFICATE OF SERVICE

Pursuant to 37 C.F.R. §§42.6(e) and 42.105, I certify that I caused to be served true and correct copies of the foregoing: **PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 11,516,231 and Exhibits 1001-1059** by Federal Express Next Business Day Delivery on November 1, 2024, on the Patent Owner's correspondence address of record for the subject patent as follows:

Finnegan, Henderson, Farabow,
Garrett & Dunner LLP
Attn: Christopher C. Johns
901 New York Avenue, N.W.
Washington, DC 20001-4413

Dated: November 1, 2024

/Matthew A. Argenti/
Matthew A. Argenti, Lead Counsel
Reg. No. 61,836