

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

DECLARATION OF DR. ANGELOS STAVROU

TABLE OF CONTENTS

I.	ENGAGEMENT.....	1
II.	QUALIFICATIONS.....	1
III.	COMPENSATION.....	3
IV.	INFORMATION CONSIDERED.....	3
V.	LEGAL PRINCIPLES.....	4
VI.	THE RELEVANT TIMEFRAME FOR ANALYSIS.....	7
VII.	PERSON OF ORDINARY SKILL IN THE ART.....	8
VIII.	STATE OF THE ART.....	8
	A. Virtualization and Cloud Computing.....	9
	B. Cyber Security.....	19
	1. Types of Security Risks.....	19
	2. Detecting Security Risks.....	26
IX.	OVERVIEW OF THE '855 PATENT.....	40
	A. Challenged Claims.....	42
	B. Prosecution History of the '855 Patent.....	44
X.	CLAIM CONSTRUCTION.....	45
XI.	OVERVIEW OF THE PRIOR ART.....	46
	A. Elder (U.S. Pub. No. 2014/0189873, EX1005).....	46
	B. Kim (U.S. Pub. No. 2016/0092679, EX1006).....	48
	C. Hufsmith (U.S. Pub. No. 2020/0097662, EX1007).....	49

XII.	GROUND 1: CLAIMS 1-6, 8-9, AND 11-18 WERE OBVIOUS OVER ELDER AND KIM	53
A.	Reasons to Combine Elder and Kim	54
B.	Independent Claims 1, 12, and 18.....	63
1.	Preambles	63
2.	Element 1.i	66
3.	Elements 1.1, 12.1, and 18.1	67
4.	Elements 1.2, 12.2, and 18.2.....	72
5.	Elements 1.3, 12.3, and 18.3.....	74
6.	Elements 1.4, 12.4, and 18.4.....	79
7.	Elements 1.5 and 12.5.....	81
8.	Elements 1.6, 12.6, and 18.5.....	83
C.	Dependent Claims	94
1.	Claims 2-3 and 13-14.....	94
2.	Claims 4 and 15.....	96
3.	Claims 5 and 16.....	98
4.	Claims 6 and 17.....	99
5.	Claim 8.....	100
6.	Claim 9.....	101
7.	Claim 11	101
XIII.	GROUND 2: CLAIMS 1-18 WERE OBVIOUS OVER ELDER, KIM, AND HUFSMITH	102
A.	Reasons to Combine Elder, Kim, and Hufsmith.....	103
B.	Claims 1, 12, and 18.....	109
C.	Claims 4 and 15.....	111
D.	Claim 7	113

E. Claim 10	114
XIV. CONCLUDING STATEMENTS.....	116
XV. APPENDIX A – MATERIALS CITED.....	118

I, Angelos Stavrou, declare as follows:

I. ENGAGEMENT

1. I have been retained by counsel for Wiz, Inc. as an expert witness in the above-captioned proceeding. I have been asked to provide my opinion about the state of the art of the technology described in U.S. Patent No. 11,637,855 (the “’855 patent”) and on the patentability of claims 1-18 of this patent. The following is my written testimony on these topics.

II. QUALIFICATIONS

2. I received my M.Sc. in Electrical Engineering, M.Phil., and Ph.D. (with distinction) in Computer Science all from Columbia University. I also hold an M.Sc. in theoretical Computer Science from the University of Athens and a B.Sc. in Physics with distinction from the University of Patras, Greece.

3. I am a Virginia Tech Innovation Campus founding Professor, and the Entrepreneurship activities lead. I am also a member of the Bradley Department of Electrical & Computer Engineering at Virginia Tech. From 2017 to 2020, I was a Professor in the Computer Science Department at George Mason University (“GMU”), teaching courses including Operating Systems Security and Cyber Security Laboratory. From 2012 to 2017, I was an Associate Professor in GMU’s Computer Science Department, teaching courses including Operating Systems Security, Enterprise Security Practices, and Enterprise Security Technology. From

2014 to 2017, I was an Academic Director in GMU's School of Management for the M.S. in Management of Secure Information Systems Program. From 2013 to 2015, I was an Academic Director in GMU's Computer Science Department for the M.S. in Information Security and Assurance Program. From 2011 to 2020, I was an Associate Researcher in the Computer Security Division at the National Institute of Standards and Technology ("NIST"). I am a co-author of numerous publications involving virtualization and cyber security, among other topics. EX1003 (CV), 3-19.

4. I am also the founder of Quokka, Kryptowire Labs, Aether Argus, and Impedyme Inc. I have served as a principal investigator on research awards from NSF, DARPA, IARPA, DHS, AFOSR, ARO, ARL, and ONR. I have written more than 140 peer-reviewed conference and journal articles. Furthermore, I am an inventor of nineteen issued patents and several pending patent applications. I am an Associate Editor of IEEE Transactions on Computers, IEEE Security & Privacy, and IEEE Internet Computing magazine and a co-chair of the IEEE Blockchain initiative. I am a senior member of the ACM, USENIX, and IEEE. My current research interests include security and reliability for distributed systems, security principles for virtualization, and anonymity, focusing on building and deploying large-scale systems.

5. I received the GMU Department of Computer Science Outstanding Research Award in 2010, 2016, and 2018. Also, I was awarded the 2012 George Mason Emerging Researcher, Scholar, Creator Award, a university-wide award. In 2013, I received the IEEE Reliability Society Engineer of the Year award. My team at Kryptowire was awarded the DHS Cyber Security Division's "Significant Government Impact Award" in 2017 and the "Bang for the Buck Award" in 2019. Currently, I am the primary Principal Investigator (PI) in two DARPA awards focusing on Cyber Security for cloud and wireless systems, namely the DARPA OPS-5G and CASTLE efforts.

III. COMPENSATION

6. I am being compensated for my time at my standard consulting rate. My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the specifics of my testimony. I have no financial interest in the outcome of this matter or in any litigation involving the '855 patent.

IV. INFORMATION CONSIDERED

7. My opinions are based on my years of education, research and experience, as well as my investigation and study of relevant materials. In preparing this declaration, I have reviewed the '855 patent and the '855 patent's file history (EX1004). I have also reviewed the materials cited in this declaration, a list of which is provided in Appendix A.

V. LEGAL PRINCIPLES

8. I understand that a claim is not patentable under 35 U.S.C. §102 as anticipated if the claim as a whole is described, either expressly or inherently, in a single prior art reference. I understand that this also requires that the prior art reference disclose that each claim element is arranged as recited in the claims.

9. I have been advised that a claimed invention is not patentable under 35 U.S.C. §103 if it is obvious. A patent claim is unpatentable if the claimed invention would have been obvious to a person of ordinary skill in the field at the time the claimed invention was made. This means that even if all of the requirements of the claim cannot be found in a single prior art reference that would anticipate the claim, a person of ordinary skill in the relevant field who knew about all this prior art would have come up with the claimed invention.

10. I have further been advised that the ultimate conclusion of whether a claim is obvious should be based upon several factual determinations. That is, a determination of obviousness requires inquiries into: (1) the level of ordinary skill in the field; (2) the scope and content of the prior art; (3) what difference, if any, existed between the claimed invention and the prior art; and (4) any secondary evidence bearing on obviousness.

11. I have been advised that, in determining the level of ordinary skill in the field that someone would have had at the time the claimed invention was made,

I should consider: (1) the levels of education and experience of persons working in the field; (2) the types of problems encountered in the field; and (3) the sophistication of the technology.

12. I have also been advised that, in determining the scope and content of the prior art, in order to be considered as prior art, a reference must be reasonably related to the claimed invention of the patent. A reference is reasonably related if it is in the same field as the claimed invention or is from another field to which a person of ordinary skill in the field would look to solve a known problem.

13. I have been advised that a patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, I may consider whether there are reasons that would have prompted a person of ordinary skill in the field to combine the elements or concepts from the prior art in the same way as in the claimed invention.

14. I have been further advised that there is no single way to define the line between true inventiveness on the one hand (which is patentable) and the application of common sense and ordinary skill to solve a problem on the other hand (which is not patentable). For example, market forces or other design incentives may be what produced a change, rather than true inventiveness. I may consider whether the

change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness. I may also consider whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent. I may consider whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way. I may also consider whether the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those skilled in the art.

15. I have also been advised, however, that I must be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact. I should put myself in the position of a person of ordinary skill in the field at the time the claimed invention was made and I should not consider what is known today or what is learned from the teaching of the patent.

16. Finally, I have been advised that any obviousness rationale for modifying or combining prior art must include a showing that a person of ordinary skill would have had a reasonable expectation of success.

17. With regard to secondary considerations of nonobviousness, I have been advised that any objective evidence may be considered as an indication that the

claimed invention would not have been obvious at the time the claimed invention was made. I understand that the purpose of secondary considerations is to prevent a hindsight analysis of the obviousness of the claims.

18. I have been advised that there are several factors that may be considered as a secondary consideration. These factors include the commercial success of the invention, industry praise for the invention, skepticism of the invention, licensing of the invention, copying of the invention, any long-felt need that the invention solved, failure of others, and unexpected results of the invention.

19. I have been further advised that in order for evidence of secondary considerations to be significant, there must be a sufficient nexus between the claimed invention and the evidence of secondary considerations. I understand that this nexus serves to provide a link between the merits of the claimed invention and the evidence of secondary considerations provided.

VI. THE RELEVANT TIMEFRAME FOR ANALYSIS

20. I understand that 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. I have been asked to treat April 26, 2021 as the effective filing date for purposes of my analysis.

VII. PERSON OF ORDINARY SKILL IN THE ART

21. As I noted earlier in §V, I have been advised that, in determining the level of ordinary skill in the field that someone would have had at the time the claimed invention was made, I should consider: (1) the levels of education and experience of persons working in the field; (2) the types of problems encountered in the field; and (3) the sophistication of the technology.

22. In my opinion, a person of ordinary skill in the art (“POSA”) at the relevant time would have had (1) at least a bachelor’s degree in computer science, computer engineering, electrical engineering, or a related degree, and (2) would also have 2-3 years of professional experience with cyber security analysis and virtualization. Such experience includes, for example, malware analysis, security analysis of cloud computing systems, and security analysis of virtual machines. I satisfied these requirements before April 26, 2021. *See supra*, §II. Additional experience can compensate for less education and vice versa. The State-of-the-Art section below summarizes some basic background knowledge of the type that persons of ordinary skill in the art would have had.

VIII. STATE OF THE ART

23. In this section, I provide a brief overview of the background art as understood by a POSA as of April 26, 2021. The material that I cite below confirms my own recollection and understanding, based on my experience in the field, of the

state of the art before April 26, 2021. First, I provide a general overview of virtualization technology and cloud computing. I then provide a high-level discussion of cyber security, including several types of threats and vulnerabilities that were commonly exploited and how those threats were detected and analyzed. I also explain how these cyber security approaches were applied in virtualized cloud computing assets available as of April 26, 2021.

A. Virtualization and Cloud Computing

24. Cloud computing typically provided on-demand availability of computer system resources (storage and compute power) from a shared pool and over a remote network. EX1008 (NIST Cloud Computing), 1 (“Cloud Computing enables ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.”).¹ To share such resources, cloud computing

¹EX1008 is a copy of a July 2013 publication by the National Institute of Standards and Technology (“NIST”), which is an agency within the U.S.

environments employed virtualization technology, which refers to the decades-old process of emulating physical computers. EX1009 (Wolf), xxiii (“In running as a virtual machine, a computer’s hardware is emulated and presented to an operating system as if the hardware truly existed.”);² EX1010 (Waldspurger), 2 (“Virtual machines have been used for decades.”)³; EX1011 (Goldberg), 35 (“[M]uch of the

Department of Commerce supporting innovation and competitiveness. This copy was obtained from <http://dx.doi.org/10.6028/NIST.SP.500-291r2>.

² EX1009 includes excerpts from “Virtualization: From the Desktop to the Enterprise,” a book by C. Wolf and E. Halter, published in 2005 by APRESS. This book was obtained from <https://link.springer.com/book/10.1007/978-1-4302-0027-7>. To avoid overburdening the record, EX1009 includes only the front matter, chapter 1, chapter 6, chapter 7, and appendix A, since the book is quite large and I only cite to chapters 1, 6, and 7 and the appendix.

³ EX1010 is a copy of “Memory Resource Management in VMware ESX Server,” by C. Waldspurger, published by USENIX in 2002 as a part of the 5th Symposium on Operating Systems Design and Implementation. This copy was

software for the simulated machine executes directly on the hardware without software interpretation. Systems of this kind are called virtual machine systems, the simulated machines are called virtual machines (VMs), and the simulator software is called the virtual machine monitor (VMM) ... IBM[] improved virtual machine support for System/370.”).⁴

25. An emulated physical computer was called a “virtual machine” or “VM.” EX1009 (Wolf), xxiii (“[W]orkstations and servers no longer need dedicated physical hardware such as a CPU or motherboard in order to run as independent entities. Instead, they can run inside a virtual machine (VM).”). A VM acted like a physical machine, in that it typically ran a standard operating system, such as Linux or Windows, and could run standard software applications, such as web servers.

obtained from

<https://www.usenix.org/legacy/event/osdi02/tech/waldspurger/waldspurger.pdf>.

⁴ EX1011 is a copy of “Survey of Virtual Machine Research,” by R. Goldberg, published by the IEEE Computer Society in June 1974 in Computer Magazine, Vol. 7, No. 6. This copy was obtained from

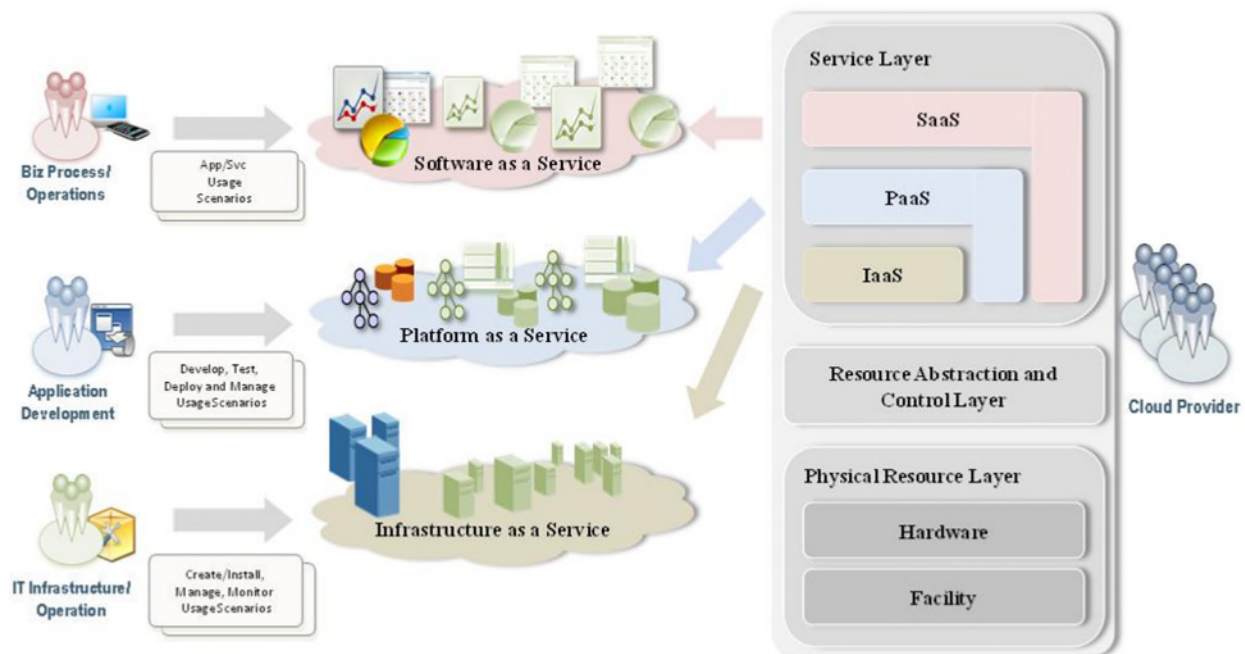
<https://ieeexplore.ieee.org/document/6323581>.

EX1010 (Waldspurger), 2 (Virtualization software such as VMware’s ESX Server “is in production use on servers running multiple instances of unmodified operating systems such as Microsoft Windows 2000 Advanced Server and Red Hat Linux 7.2.”). Software containers were another form of virtualization that were well understood and widely used at the time. EX1012 (NIST Application Container), ii (“Application container technologies, also known as containers, are a form of operating system virtualization combined with application software packaging. Containers provide a portable, reusable, and automatable way to package and run applications. This publication explains the potential security concerns associated with the use of containers and provides recommendations for addressing these concerns.”);⁵ EX1013 (Rosendahl), 1:16-20 (“In the container model, a kernel of an operating system (e.g., Linux) allows for multiple isolated user-space instances, or ‘containers,’ executing simultaneously.”); *see also id.*, 1:20-35 (discussing operation and benefits of containers). Desktop, server, cloud, and datacenter providers

⁵ EX1012 is a copy of a September 2017 NIST publication obtained from <https://doi.org/10.6028/NIST.SP.800-190>.

routinely used many different virtualization solutions and products, spanning both closed and open-source, available for desktop, server, and cloud applications.

26. Cloud computing services were commonly built on a three-layered framework: a service layer, a resource abstraction layer, and a physical resource layer. This three-layered framework enabled three main types of service model: SaaS (software as a service), PaaS (platform as a service), and IaaS (infrastructure as a service). The figure below illustrates a cloud computing environment:



EX1008 (NIST Cloud Computing), 18.

27. “The top layer is the service layer, where a cloud provider defines and provisions each of the three service models. This is where cloud consumers consume cloud services through the respective cloud interfaces.” EX1008 (NIST Cloud

Computing), 19. In the SaaS model, applications were accessible from various client devices through the service layer, either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. Cloud providers also offered VMs as cloud instances that a user could access through APIs. *See, e.g.*, EX1014 (VMware Public Cloud), 29;⁶ EX1015 (VMware vSphere), 55-58, 62-66;⁷ EX1008

⁶EX1014 is a copy of “Public Cloud Service Definition,” by A. Roberts, published in January 2018 by VMware, Inc. This copy was obtained from <https://www.vmware.com/content/dam/digitalmarketing/vmware/en/pdf/vcat/vmware-public-cloud-service-definition.pdf>.

⁷ EX1015 is a copy of the vSphere Web Services SDK Programming Guide, published in April 2018 by VMware, Inc. This copy was obtained from <https://vdc-download.vmware.com/vmwb-repository/dcr-public/cdbbd51c-4824-4a1b-ad43-45df55a76a76/8cb3ed93-cac2-46aa-b329-db5a096af5bc/vsphere-web-services-sdk-67-programming-guide.pdf>. To avoid overburdening the record, EX1015 includes only the front matter, chapter 2, chapter 4, chapter 5, chapter 8, chapter 9, chapter 10, and chapter 11, since the document is quite large and I only cite to chapters 2, 4, 5, 8, 9, 10, and 11.

(NIST Cloud Computing), 94-95. Through APIs, GUIs, or other consoles, a user could control and manage their cloud resources, for example, by sending queries and commands to a hypervisor. EX1015 (VMware vSphere), 164-66, 118 (Fig 8-2), 138 (Fig. 9-1).

28. “The middle layer is the resource abstraction and control layer.... The layer typically includes software elements such as hypervisors, virtual machines, virtual data storage, and other resource abstraction and management components needed to ensure efficient, secure, and reliable usage.” EX1008 (NIST Cloud Computing), 19. With the interfaces provided by the service layer, a user could interact with the abstraction layer to, for example, spin up virtual machines, allocate virtual storage, and perform data backups. Virtualization technology (including virtual machines and virtual disk storage) was thus central to cloud computing.

29. “The lowest layer in the framework is the physical resource layer, which includes all the physical computing resources.” EX1008 (NIST Cloud Computing), 19. The physical layer’s resources (including servers, physical disk storage, and network components) were frequently concentrated in one or more “data centers.” EX1016 (Wood), 229 (“Data centers—server farms that run

networked applications—have become popular in a variety of domains such as web hosting, enterprise systems, and e-commerce sites.”).⁸

30. Furthermore, off-the-shelf virtualization tools (e.g., VMWare® VirtualCenter) routinely used APIs to query for information about a target VM. These API tools would often provide a graphical user interface (“GUI”) providing various virtual-resource-management features such as allowing users to view their VMs and associated information. For example, it was commonplace for GUIs to allow users to select a particular VM, at which point more detailed information about the VM (e.g., the names and locations of associated virtual disks) would be presented to the user based on API calls. EX1017 (VirtualCenter), 445-46 (interface

⁸ EX1016 is a copy of “Black-box and Gray-box Strategies for Virtual Machine Migration,” by T. Wood et al., published by USENIX in 2007 as a part of NSDI ’07: 4th USENIX Symposium on Networked Systems Design & Implementation. This copy was obtained from <https://www.usenix.org/conference/nsdi-07/black-box-and-gray-box-strategies-virtual-machine-migration>.

displaying information for a virtual machine and its resources);⁹ EX1018 (Price), ¶¶21 (“Virtual machine manager 130 interfaces can include interfaces and application programming interfaces (APIs) that can provide ... virtual disk access....”), 42 (“[S]ecurity server 305 can use a virtual disk API of the virtual machine manager 330 to mount a virtual image of an offline virtual machine and read files of the offline virtual machine ... The security server 305 can thereby collect registry, file, and policy setting values from the images and files obtained over the virtual machine manager 330 API(s) and then analyze these values for vulnerabilities....”).

31. Furthermore, this type of API-based communication was not limited to UIs. Common cloud/virtualization platforms included APIs that were routinely used by GUIs and other types of systems for querying information about virtual resources.

⁹ EX1017 is a copy of “VMware VirtualCenter User’s Manual version 1.4,” published by VMware, Inc. on July 7, 2006. This copy was obtained from https://www.vmware.com/pdf/vc_users14.pdf. To avoid overburdening the record, EX1052 includes only the front matter, chapter 8, and chapter 17, since the document is quite large and I only cite to chapters 8 and 17.

For example, in the vSphere platform from VMware®, the “find_entity_views()” API call would provide the inventory of the user’s VMs, including allocated virtual disks, and “get_inventory_path()” would provide locations of the identified disk(s). EX1019 (Virtual Disk),¹⁰ 13, 23, 32-33, 53-56, 68-69. These API calls, whether initiated through the use of a GUI or another programming interface, were routinely used as building blocks of more complex tasks, such as displaying summary information to a user about a target VM (EX1017 (VirtualCenter), 445-46) or performing a virus scan of the virtual disks of a target VM (EX1019 (Virtual Disk), 55-56).

¹⁰EX1019 is a copy of “VMWare Virtual Disk Programming Guide,” published by VMware, Inc. on March 25, 2016. This copy was obtained from https://vdc-download.vmware.com/vmwb-repository/dcr-public/ae1a858c-2f59-4a75-b333-f2ddfa631fbd/4ae542fa-c7ec-4713-a3a0-6fef938b03d1/vddk60_programming.pdf. To avoid overburdening the record, EX1055 includes only the front matter, chapter 1, chapter 3, chapter 4, chapter 6, and chapter 7, since the document is quite large and I only cite to chapters 1, 3, 4, 6, and 7.

B. Cyber Security

32. Cyber security has long been an important topic in IT applications. Below, I first describe general threats to computing resources, how these threats apply to virtual machines, and how they appear in the cloud. Next, I discuss how these threats and vulnerabilities were often detected, the common tools used to do so, and how detected threats and vulnerabilities were subsequently analyzed.

1. Types of Security Risks

33. It was well known that computer systems faced various types of security risks, including threats that took advantage of vulnerabilities in the computer system. EX1020 (NIST Security Patches), 2 (“Vulnerabilities are weaknesses in software that can be exploited by a malicious entity to gain greater access and/ or permission than it is authorized to have on a computer.”).¹¹ For example, computer systems were often targeted with “malicious code and malicious software that are covertly inserted into an information technology (IT) system to compromise the confidentiality, integrity, or availability of the data, applications, or

¹¹EX1020 is a copy of an October 2002 NIST publication obtained from https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=150702.

operating system, or to annoy or disrupt the system’s owner,” commonly referred to as malware. EX1021 (NIST Preventing Malware), 1.¹² Old, outdated software often contained bugs that, if not patched, could be exploited. EX1020 (NIST Security Patches), 1 (“Despite extensive testing, all operating systems and applications are released with bugs (errors in the software) that affect security, performance, and stability.... Many security related bugs are generally discovered only after a large number of users start using the software, and hackers and independent testers start attempting to compromise it. Once a bug is discovered, the software manufacturer often releases a piece of software to correct the bug. This software is often called a patch, hotfix, or service pack.”).

34. Many vulnerabilities were catalogued by individuals and organizations in the field and published as a set of Common Vulnerabilities and Exposures (“CVE”) to facilitate identification and mitigation of these known threats. EX1020 (NIST Security Patches), 1-3 (“Technical vulnerability experts from 31 industry,

¹²EX1021 is a copy of a December 2005 NIST publication obtained from <https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul2005-12.pdf>.

academia, and government organizations vote on the common names. CVE provides the computer security community with: a comprehensive list of publicly known vulnerabilities, an analysis of the authenticity of newly published vulnerabilities, and a unique name to be used for each vulnerability.”). Cyber security assessments were also concerned with vulnerabilities that were unknown at the time. For example, computers were often scanned to detect potential problems related to unknown software vulnerabilities so that these vulnerabilities could be better understood and resolved. EX1022 (NIST Incident Handling), 12 (“More information may need to be provided in some cases—for example, if a server appears to have been compromised through an unknown software vulnerability.”).¹³ The European Union Agency for Network and Information Security (ENISA) “Threat Landscape Report” for 2018 shows that the top threats and vulnerabilities were from attacks external to the network, including malware and web-based threats, as illustrated below:

¹³EX1022 is a copy of an August 2012 NIST publication obtained from <http://dx.doi.org/10.6028/NIST.SP.800-61r2>.

Top Threats 2017	Assessed Trends 2017	Top Threats 2018	Assessed Trends 2018	Change in ranking
1. Malware	↔	1. Malware	↔	→
2. Web Based Attacks	↑	2. Web Based Attacks	↑	→
3. Web Application Attacks	↑	3. Web Application Attacks	↔	→
4. Phishing	↑	4. Phishing	↑	→
5. Spam	↑	5. Denial of Service	↑	↑
6. Denial of Service	↑	6. Spam	↔	↓
7. Ransomware	↑	7. Botnets	↑	↑
8. Botnets	↑	8. Data Breaches	↑	↑
9. Insider threat	↔	9. Insider Threat	↓	→
10. Physical manipulation/ damage/ theft/loss	↔	10. Physical manipulation/ damage/ theft/loss	↔	→
11. Data Breaches	↑	11. Information Leakage	↑	↑
12. Identity Theft	↑	12. Identity Theft	↑	→
13. Information Leakage	↑	13. Cryptojacking	↑	NEW
14. Exploit Kits	↓	14. Ransomware	↓	↓
15. Cyber Espionage	↑	15. Cyber Espionage	↓	→

Legend: Trends: ↓ Declining, ↔ Stable, ↑ Increasing
Ranking: ↑ Going up, → Same, ↓ Going down

Table 1- Overview and comparison of the current threat landscape 2018 with the one of 2017

EX1023 (ENISA Report, 2018), 9.¹⁴

35. Cloud-based resources, such as virtual machines, often had the same vulnerabilities as traditional physical machines, as well as additional vulnerabilities

¹⁴EX1023 is a copy of the “ENISA Threat Landscape Report 2018,” published in January 2019 by the European Network and Information Security Agency (“ENISA”). This copy was obtained from

<https://www.enisa.europa.eu/publications/enisa-threat-landscape-report-2018>.

related to the virtualization itself. EX1024 (NIST Full Virtualization Technologies), 3-1 (“Migrating computing resources to a virtualized environment has little or no effect on most of the resources’ vulnerabilities and threats. For example, if a service has certain vulnerabilities and that service is moved from a non-virtualized server to a virtualized server, the service is still just as vulnerable to exploitation. However, the use of virtualization may help reduce the impact of such exploitation—but virtualization may also provide additional attack vectors, thus increasing the likelihood of successful attack.”).¹⁵ For example, virtualization often added complexity in the form of a “host computer” and a “hypervisor” to manage the virtual machine, with each component potentially having its own vulnerabilities. *Id.*, ES-2 (“The security of the entire virtual infrastructure relies on the security of the virtualization management system that controls the hypervisor and allows the operator to start guest OSs, create new guest OS images, and perform other actions.”), 4-1 (same), 4-3 (“Because hosted virtualization systems are run under host OSs, the security of every guest OS relies on the security of the host OS.”).

¹⁵ EX1024 is a copy of a January 2011 NIST publication obtained from <https://doi.org/10.6028/NIST.SP.800-125>.

36. It was also well known at the time that cloud computing systems, including the data centers and virtual machines used in cloud systems, were common targets of cyber attacks. For example, Verizon's Data Breach Investigations Report for 2019 shows that servers were the number one asset to be targeted by hacking and malware, as illustrated below:

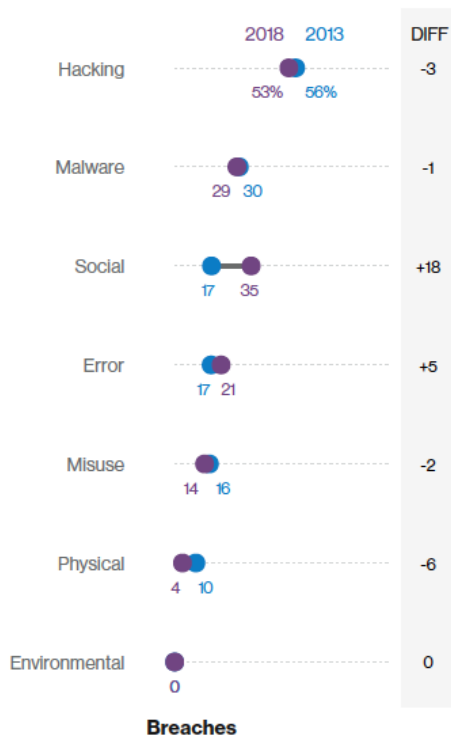


Figure 9. Threat actions in data breaches over time
n=2,501 (2013), n=1,638 (2018)

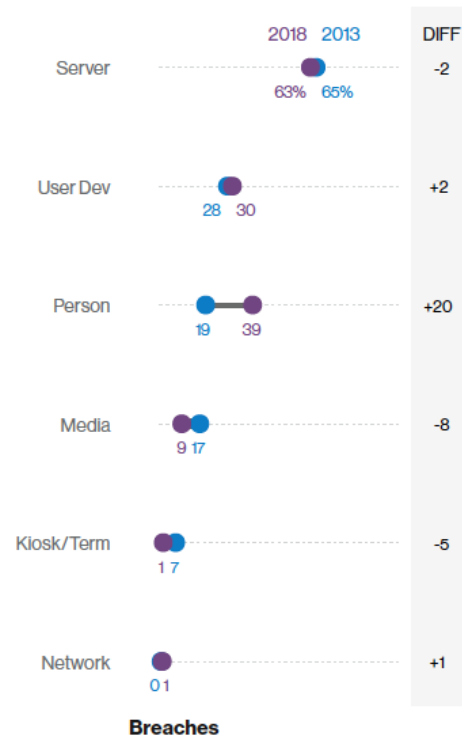


Figure 10. Asset categories in data breaches over time
n=2,294 (2013), n=1,513 (2018)

EX1025 (Verizon DBIR), 8.¹⁶ Cloud computing systems were known to have many of the same vulnerabilities as non-cloud systems, while also presenting additional risks and vulnerabilities. EX1026 (Di Pietro), 171-72.¹⁷ For example, the National Institute of Standards and Technology noted numerous security challenges facing cloud systems, including:

- Compromises to the confidentiality and integrity of data in transit to and from a cloud provider and at rest;
- Attacks which take advantage of the homogeneity and power of cloud computing systems to rapidly scale and increase the magnitude of the attack;

¹⁶EX1025 is a copy of the 2019 Data Breach Investigations Report, summarizing data from 2018, published by Verizon in May 2019. This copy was obtained from <https://www.verizon.com/business/resources/reports/2019-data-breach-investigations-report.pdf>.

¹⁷EX1026 is a copy of “Virtualization Technologies and Cloud Security: Advantages, Issues, and Perspectives” by R. Di Pietro and F. Lombardi, published as a chapter in the book “From Database to Cyber Security” in November 2018. This copy was obtained from https://doi.org/10.1007/978-3-030-04834-1_9.

- A consumer's unauthorized access (through improper authentication or authorization, or exploit of vulnerabilities introduced maliciously or unintentionally) to software, data, and resources provisioned to, and owned by another authorized cloud consumer;
- Increased levels of network-based attacks that exploit software not designed for an Internet-based model and vulnerabilities existing in resources formerly accessed through private networks;
- ...
- Attacks that exploit inconsistencies in global privacy policies and regulations....

EX1008 (NIST Cloud Computing), 45-46. Others noted that cloud virtual machines frequently encountered malware injection attacks. *See, e.g.*, EX1027 (Singh), 321 (“A first considerable attack attempt aims at injecting a malicious service implementation or virtual machine into the Cloud system. Such [c]loud malware could serve any particular purpose the adversary is interested in, ranging from eavesdropping via subtle data modifications to full functionality changes or blockings.”).

2. Detecting Security Risks

37. Users and administrators routinely sought to identify the types of security risks described above using various security analysis techniques. One common approach was to use software to scan computer systems for security vulnerabilities. *See* EX1028 (Bairwa), 116 (“Vulnerability assessment means

identifying the vulnerabilities in the system before they could be used by anyone else with bad intentions of harming the network.... Vulnerability scanners are the ones that find out the vulnerabilities in the system which if accessed by a malicious user or hacker can put the whole network system at risk.”). Such scanners were routinely used to evaluate operating systems, applications, files, configurations, and other aspects of computer systems (both physical and virtual) to identify potential vulnerabilities. See EX1029 (NIST IT Asset Mgmt), 22;¹⁸ EX1030 (Li), ¶¶27, 43; EX1033 (Garfinkel), 2;¹⁹ EX1028 (Bairwa), 116; EX1032 (Burdett), ¶43 (“The threat management facility 100 may provide configuration management, which may

¹⁸EX1029 is a copy of a September 2018 NIST publication obtained from <http://doi.org/10.6028/NIST.SP.1800-5>.

¹⁹EX1033 is a copy of “A Virtual Machine Introspection Based Architecture for Intrusion Detection,” by T. Garfinkel and M. Rosenblum, published in February 2003 as a part of the Network and Distributed System Security (NDSS) 2003 Symposium. This copy was obtained from <https://www.ndss-symposium.org/ndss2003/virtual-machine-introspection-based-architecture-intrusion-detection/>.

be similar to policy management, but may specifically examine the configuration set of applications, operating systems, hardware, and the like, and manage changes to their configurations. Assessment of a configuration may be made against a standard configuration policy, detection of configuration changes, remediation of improper configuration, application of new configurations, and the like.”).

38. In the virtualization context, it was well known that security scans could be performed in a variety of different ways and on different types of data. For example, scanners were sometimes installed on the virtual machine as an agent that would then run as an application on the targeted system, performing its scans while the system is in operation. *See, e.g.*, EX1034 (Tang), 5643 (“In the internal approach, antivirus software is installed in the VM and it often uses signature-matching technology to search viruses in files.”). Although directly inspecting the state of VMs with an agent provided high visibility, this visibility frequently came at the cost of weaker isolation, meaning that an attacker who successfully infected a guest VM’s operating system could overpower the agent. EX1033 (Garfinkel), 1; EX1037 (Bauman), 10:5 (discussing the advantages of agentless VM monitoring); *see also* EX1032 (Burdett), ¶¶5-6 (noting desirability of performing security analysis “without use of an agent installed on the template image used in cloud-based environments”).

39. Another common approach involved external scanning software that was not run as an application on the scanned system. As far back as 2003, the technique of Virtual Machine Introspection (VMI), also known as “external” or “out-of-VM” security, became an increasingly common approach involving scanning software located in the hypervisor that was not run as an application or agent on the scanned guest VM. *E.g.*, EX1033 (Garfinkel), 1; EX1034 (Tang), 5643 (“The external approach is inspired by the feature that the VMM has supervisory privilege on guest VMs. Antivirus software is totally moved out of the VM, and gets necessary information from the guest OS using VMI techniques....”); *see also* EX1034 (Tang), 5643-644 (discussing hybrid approach); EX1035 (Lengyel), 389 (distinguishing its techniques from “prior agentless VMI approaches”).²⁰ For example, scanners often evaluated a virtual machine and its data in native format,

²⁰EX1035 is a copy of “Scalability, Fidelity and Stealth in the DRAKVUF Dynamic Malware Analysis System,” by T. Lengyel et al., published in December 2014 as a part of ACSAC '14: Proceedings of the 30th Annual Computer Security Applications Conference. I obtained this copy from <https://dl.acm.org/doi/pdf/10.1145/2664243.2664252>.

such as an image file. EX1036 (Wysopal), 3:56-65 (“The analysis may also include scanning the executing image file (and, in some cases, executing applications within the virtual machine) using a network vulnerability scanner and/or a host vulnerability scanner.”); *see also* EX1024 (NIST Full Virtualization Technologies), 3-3 (“Image files can also be scanned to detect rootkits and other malware that, when running, conceal themselves from security software present within the guest OS.”). By 2015, VMI had developed into a well-established field, with a robust literature and a wide array of techniques. *See, e.g.*, EX1037 (Bauman), 10:9-10 (surveying 64 different out-of-VM approaches); EX1038 (Suneja), 133-34 (creating a taxonomy to summarize the field of VMI). VMI has been used to support a wide range of use cases, including kernel integrity and security monitoring, intrusion detection, anti-malware, firewall, virus scanning, cloud-wide information flow tracking, and policy enforcement. EX1038 (Suneja), 133.²¹

²¹EX1038 is a copy of “Exploring VM Introspection: Techniques and Trade-offs,” by S. Suneja et al., published in March 2015 as a part of VEE '15: Proceedings of the 11th ACM SIGPLAN/SIGOPS International Conference on

40. Scanners typically evaluated the computer system to identify different types of vulnerabilities. For example, scanners often checked the configuration files of operating systems and installed applications, looking for out-of-date software that needed to be patched. EX1039 (Giakouminakis), 3:50-54 (“For example, a vulnerability in the software resources can include, for example, software that is out of date, software that has known security weakness, configurations of software that have known security weaknesses, known bugs of software, etc.”); EX1040 (Carroll), 31-36 (discussing use of known logs and configuration files used in forensic analysis). Scanners also routinely looked for signatures of known malicious code. EX1039 (Giakouminakis), 5:28-32 (“In embodiments, when the vulnerability is identified, the security tool 102 can identify whether the identified vulnerability is associated with known malware and/or is associated with exploit kits used for the creation of malware.”). Matching system data against public repositories was a common way of detecting security issues such as applications (or versions of applications) known to be vulnerable or files infected with malware. EX1041

Virtual Execution Environments. This copy was obtained from

<https://dl.acm.org/doi/abs/10.1145/2817817.2731196>.

(Milner), 2:24-67 (“To check the security, the security tool can be configured to compare the components of the software package to the black list. [...] A black list can include a list of archival files that are known to be insecure, such as known insecure packages/subpackages referenced in a Common Vulnerabilities and Exposures (CVE) list.”); EX1042 (Khalid), 2:45-51 (“Moreover, an object engine may be configured to observe and/or extract characteristics of a suspicious object and compare such observed characteristics with those stored on one or more cyber-security databases, such as, for example, a Common Vulnerabilities and Exposures (CVEs) database.”), 8:1-30 (discussing CVE databases and other cyber-security databases); EX1059 (Hibbert), ¶¶41-51, 56 (examining file system, registry and configuration files to identify installed application versions and comparing to a database of known vulnerabilities). Various types of matching techniques were well known and routinely used. *E.g.*, EX1047 (Bourquin), 1-5, 7 (discussing known binary comparison tools such as BinDiff for identifying malware); EX1043 (Epstein), 14:58-15:18 (discussing “byte-by-byte comparison”); EX1044 (Czarny), 3:56-4:24 (discussing matching hash of software product with hash known vulnerability), 11:27-47 (discussing performing binary comparison with binary of known vulnerability).

41. Scanners were also commonly used to identify and classify important assets to help prioritize risks. EX1039 (Giakouminakis), 6:9-18 (“Assets can be assigned a classification that will automatically assign specific values that drive the risk associated with vulnerabilities detected on that asset. The classification can describe the importance of the asset (e.g., importance of confidentiality, integrity, or availability) to the entity controlling the asset, the cost to mitigate vulnerabilities, and/or the cost to deal with consequences of an attack on the asset. In embodiments, in determining the risk level, the security tool 102 can be configured to utilize an additional factor that represents the classification of the asset.”); EX1045 (Mohanty), 1:5-22 (“Consequently, in virtualized environments, any risks, threat exposures or known vulnerabilities are constantly changing. A security operations team needs a strong and continuous prioritization system to track critical vulnerabilities and take actions as changes occur.”). Security risks were routinely prioritized based on multiple factors, including factors based on the network accessibility of the vulnerability or asset. *E.g.*, EX1045 (Mohanty), 1:23-42 (discussing “common vulnerability scoring system” or “CVSS”), 4:24-5:15 (vulnerability-assessment products “known as scanners scan virtual machines and report vulnerabilities with additional information such as Severity, Base CVSS Score 222, Temporal CVSS Score 222, and CVE ID 220”), 5:16-58 (discussing consideration of “workload

context”), 5:59-6:44 (prioritization score based on vulnerability/threat/workload scores and other factors), Figs. 2-4; EX1046 (Hovor), ¶¶48 (“The system may determine priority information for an asset, type of data store on an asset, or both, using the location of the asset in the network topology.”), 75-77 (asset score based in part on asset scores of other connected assets).

42. For example, vulnerabilities often existed in software applications installed on an asset. EX1048 (Basavapatna), ¶¶37 (identifying “particular software products executing on the system (e.g., a particular word processor or a particular web server)” as part of vulnerability analysis), 44, 53, 60, 79-80; EX1039 (Giakouminakis), 3:50-57. Identifying vulnerabilities contained in installed software applications was important because an attack from outside the asset could “occur[] when the unauthorized entity exploits a known vulnerability of the asset in an attempt to access or control the asset.” EX1048 (Basavapatna), ¶3; *see also* EX1039 (Giakouminakis), 1:18-31. Thus, POSAs understood that by identifying ways in which an attacker could exploit a software vulnerability, an administrator could gain a better understanding of the most at-risk vulnerabilities and assets, allowing the administrator to better focus remediation efforts. EX1048 (Basavapatna), ¶¶21 (describing active countermeasures, such as “applying a patch to a vulnerable application or OS component [that] removes the vulnerability”), 22-

25 (describing passive countermeasures, such as “leveraging a firewall to protect a port that is being listened to by an application that is vulnerable”), 29; EX1045 (Mohanty), 3:28-42 (identifying “criticality of vulnerability” important for security teams in assessing vulnerabilities that “may require immediate attention”).

43. Information known to be useful in providing this insight was the type of network accessibility needed to exploit a known vulnerability. For example, risk analysis systems often made use of the Common Vulnerability Scoring System (“CVSS”), which was “an open framework for communicating the characteristics and severity of software vulnerabilities.” EX1049 (CVSS Version 3), 3;²² *see also*

²² EX1049 is a copy of “Common vulnerability scoring system 3.0” (ITU-T X.1521) obtained from https://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-X.1521-201603-I!!PDF-E&type=items. It discusses a standard provided by the International Telecommunication Union (an agency of the United Nations) regarding the CVSS framework. The document indicates that it is a version from March 2016, and it confirms my own understanding and recollections regarding aspects of CVSS scores that would have been well known to POSAs prior to April 2021.

EX1048 (Basavapatna), ¶55; EX1039 (Giakouminakis), 4:36-50 (determining risk level based on CVSS score); EX1045 (Mohanty), 3:50-54 (risk analysis based on base/temporal CVSS score); EX1007 (Hufsmith), ¶¶116-17; EX1056 (Rapid7), 2 (CVSS “helpful in telling organizations which vulnerabilities they should focus on first”).²³ Each known software vulnerability (identified by a CVE identifier) was associated with a CVSS score that included a numerical score and a vector string used to derive the score. EX1049 (CVSS Version 3), 3; EX1031 (NIST NVD Overview), 1 (“NIST provides ongoing analysis of Common Vulnerabilities and Exposures (CVEs) and assigns Common Vulnerability Scoring System (CVSS) base metrics for each vulnerability. Ongoing analysis and scoring helps NVD users to understand the potential severity of each issue, and helps users to prioritize

²³ EX1056 is a copy of “Vulnerability Management and Scanning: Monitor networks to keep vulnerabilities in check,” archived on July 28, 2018, by the Internet Archive’s “Wayback Machine” at <https://web.archive.org/web/20180728052753/www.rapid7.com/fundamentals/vulnerability-management-and-scanning/>.

vulnerability management activities.”).²⁴ The vector string was comprised of a number of metrics that informed the exploitability of the particular vulnerability. EX1049 (CVSS Version 3), 4-5, Fig. 1; *see also* EX1050 (CVSS Version 2), 7.²⁵ For example, the base metrics of a CVSS score could inform an administrator of the type of network conditions that were required to exploit a vulnerability, such as whether an attacker needed local or adjacent network access or could exploit the vulnerability remotely. EX1049 (CVSS Version 3), 6, Table 1; *see also* EX1050 (CVSS Version 2), 7; EX1007 (Hufsmith), ¶117. The CVSS score metrics could

²⁴ EX1031 is a copy of a December 2013 NIST publication obtained from <https://www.nist.gov/publications/national-vulnerability-database-nvd-overview>.

²⁵ EX1050 is a copy of “A Complete Guide to the Common Vulnerability Scoring System Version 2.0” obtained from https://tsapps.nist.gov/publication/get_pdf.cfm?pub_id=51198. It discusses version 2 of the CVSS framework. The document indicates that it is a version from July 2007, and it confirms my own understanding and recollections regarding aspects of CVSS scores that would have been well known to POSAs prior to April 2021.

also inform the administrator of the type of access conditions needed and privileges required to successfully exploit the vulnerability. EX1049 (CVSS Version 3), 7, Tables 2-3; *see also* EX1050 (CVSS Version 2), 8-9; EX1007 (Hufsmith), ¶117.

44. Another type of useful information was the network port settings for a given asset. EX10X48 (Basavapatna), ¶37 (obtaining “particular network port settings (e.g., network port 8 is open)” as part of asset configuration data for risk analysis); EX1006 (Kim), ¶15 (obtaining “network port status information of the virtual machine” for vulnerability analysis); EX1056 (Rapid7), 2 (identifying open port settings for vulnerability analysis). It was known that software applications commonly communicated via open ports to receive data. EX1051 (Nyanchama), ¶¶5 (“To facilitate the receipt of data for multiple processes or applications, it is common to employ data ports in the transport layer”), 7. It was also known, however, that open ports provided a means for an attacker to attack vulnerable applications bound to such ports. *Id.*, ¶¶6-8; *see also* EX1052 (Cole), ¶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.”); EX1057 (Murthy), ¶¶39 (mitigating risk of outside attack by “block[ing] network access to a port used by a vulnerable program”), 57 (same); EX1058 (Murchison), 4:10-36 (explaining attackers identify accessible ports to target vulnerable software

applications); EX1048 (Basavapatna), ¶22 (mitigating outside attack risk by “leveraging a firewall to protect a port that is being listened to by an application that is vulnerable”). Thus, knowledge of the status of a port (e.g., open or closed) on an asset could inform whether an outside attack could be accomplished.

45. It was also commonplace for risk analysis assessments to consider additional environment-specific information in assessing the degree to which a vulnerable application present on an asset could be attacked, such as network location and accessibility related to the asset itself. EX1048 (Basavapatna), ¶¶15 (incorporating “an environmental component” into risk analysis that considered “features of a particular system including the particular asset”), 22, 55-56; EX1045 (Mohanty), 3:43-47 (considering “workload context” of asset as part of risk analysis, including “whether an asset is external facing, Web-connected, has sensitive data, location, etc.”), 5:16-6:1. For example, whether an asset has Web-connectivity or the asset’s placement relative to other assets in the network were considerations known to provide additional context of the risk that a particular software vulnerability posed to potential attack. EX1045 (Mohanty), 4:41-46, 8:47-50 (“A vulnerability that relies on access via the Web, and an asset that has Web-connectivity, would generate a higher contextual score 216 than would be the case for an asset that does not have Web-connectivity.”); *see also* EX1048 (Basavapatna),

¶¶59 (performing network traffic checks), 66 (consideration of passive countermeasures including network-based sensors, such as firewalls, as part of risk analysis); EX1039 (Giakouminakis), 5:4-22 (determining whether vulnerabilities are associated with a VM that may lead to exploitation of a hypervisor vulnerability), 8:46-9:3 (risk analysis for specific asset may consider impact of vulnerabilities contained in connected assets); EX1007 (Hufsmith), ¶¶92, 140-41; EX1059 (Hibbert), ¶¶76-81 (considering network access rights of user of vulnerable application).

IX. OVERVIEW OF THE '855 PATENT

46. The '855 Patent is entitled “Systems and Methods for Managing Cyber Vulnerabilities.” EX1001, Face. The '855 Patent discloses techniques “for securing cloud infrastructure to help meet compliance mandates, without disrupting business operations in live environments.” EX1001, 1:15-18. As part of these techniques, the '855 patent describes “a method of operating a cybersecurity system performing a side scanning function to protect against potential vulnerabilities,” as illustrated in Figure 5, reproduced below. *Id.*, 22:31-33, Fig. 5.

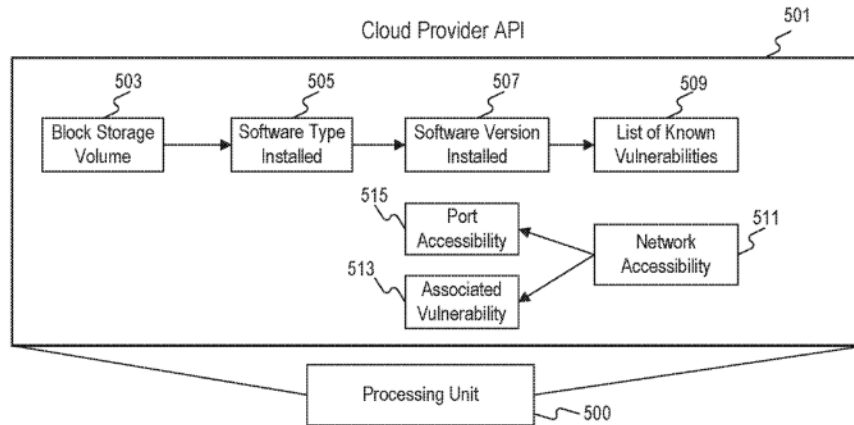


FIG. 5

Id., Fig. 5.

47. A “processor unit 500 may be communicatively connected with a cloud provider API 501” to “scan for network access or security information of a block storage volume 503 to detect potential vulnerabilities.” *Id.*, 22:35-45. Based on the accessed block storage volume 550, the processor unit 500 “may perform an identification of the type of software installed 505” and “may then identify an installed version 507 of said installed software 505.” *Id.*, 22:46-54; *see also id.*, 23:36-55. “[A] list of known vulnerabilities 509” of the installed software version 507 of the installed software 505 is then provided to the scanning system 101. *Id.*, 22:55-60; *see also id.*, 23:56-24:18 (“performing a lookup of the identified installed software version” in a data structure of software vulnerabilities “to identify known

vulnerabilities”). “Based on the list of known vulnerabilities 509,” the system may then assess the risk that an identified vulnerability can be exploited by identifying “one or more ports of accessibility to said block storage volume 503 by a known, associated vulnerability 513 from listed vulnerabilities 509 to determine an avenue for potential vulnerability 513 to access and infiltrate block storage volume 503.” *Id.*, 22:60-67; *see also id.*, 24:19-35, 24:47-62, 32:12-26 (embodiments “may include using the network accessibility information 511 and the identified at least one port 515 to identify one or more vulnerabilities 513 susceptible to attack from outside the workload”).

A. Challenged Claims

48. The '855 patent has 18 claims, with claims 1, 12, and 18 being independent. I understand that each of claims 1-18 is being challenged in the *inter partes* review proceeding related to this declaration. Below, I reproduce the text of independent claim 1 with numbering added to certain claim elements for ease of reference.

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.

49. Furthermore, independent claim 12 is directed to a method, and independent claim 18 is directed to a computer-readable medium, but these claims essentially mirror claim 1. The dependent claims add other conventional aspects of cybersecurity and cloud computing, including implementing a remedial action in response to the identified vulnerabilities (claims 2 and 13), transmitting an alert as the remedial action (claims 3 and 14), examining data sources associated with the workload as part of determining network accessibility information (claims 4 and 15), extracting data from operating system packages, libraries, or program language libraries to identify the installed software application (claims 5 and 16), configuring

a processor to identify a version of the installed software application (claims 6 and 17), identifying an installed software application that includes one or more scripts (claim 7), including aggregated vulnerability data in the data structure (claim 8), including aggregated vulnerability data from one or more third-party vendors (claim 9), including aggregated vulnerability data from data collected by a scanner (claim 10), and including aggregated vulnerability data from at least one of an advisory, an exploit, a security announcement, or a known bug (claim 11).

B. Prosecution History of the '855 Patent

50. I understand that the claims of the '972 application (i.e., the patent application that led to the '855 patent) were amended in response to rejections made by the Examiner. I understand that the last limitation of the independent claims originally recited the performance of at least one of three actions: (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. I understand that the Examiner rejected these independent claims as anticipated by Dye (U.S. Pub. No. 2020/0389482), asserting Dye taught the first action—i.e., query the cloud provider API to determine network accessibility information related to the workload. EX1004, 580-82, 1030-31.

51. I further understand that, in response, the applicant amended the claims to remove the first action from the claims, but the Examiner responded that Stopel (U.S. Pub. No. 2019/0116199) taught the second action—i.e., identify at least one port on which the vulnerable application is accessible. EX1004, 548, 563. I understand the applicant then removed the second action from the independent claims and amended the third action to 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.” EX1004, 461-62. I understand the applicant argued that Stopel did not teach this amended limitation because “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.’” EX1004, 468. I also understand the Examiner subsequently allowed the application. EX1004, 16-25.

X. CLAIM CONSTRUCTION

52. I understand that, in this proceeding, the claims are construed according to their ordinary and customary meaning, in view of the specification and

prosecution history, to a POSA as of the effective filing date of the underlying patent application. I understand that this is the same claim construction approach used in district court litigation. I have followed this approach in my analysis below.

XI. OVERVIEW OF THE PRIOR ART

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

53. Elder describes systems and methods for automated risk analysis of host devices in a network environment, such as a cloud computing environment, that uses host configuration data and vulnerability information to assess and prioritize cyber security risk. EX1005 (Elder), 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 (Kim), ¶10.

54. More specifically, Elder’s risk assessment first involves accessing host configuration information of a host. EX1005 (Elder), Abstract, ¶¶7, 28, 57, Fig. 7. The host configuration information includes “information about a plurality of software applications and an operating system on the host,” including their associated versions. *Id.*, ¶¶7, 28 (collected data “includes configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each”), 33-34, 37-38, Fig. 3. The system then queries a

vulnerability database of known vulnerabilities associated with specific versions of software applications and, based on the host configuration information, identifies vulnerabilities that match the software applications and associated versions corresponding to the host. *Id.*, ¶¶7, 29 (risk analysis module “maintains an updated database of vulnerabilities (e.g., such as that provided by the National Vulnerability Database (NVD), fix databases, etc.)”), 30 (system “correlates the host configuration data against the vulnerability database data”), 32, 39 (vulnerability database configured to report matched vulnerabilities that include “a plurality of common vulnerability enumerations (CVEs)”), 58-59, Figs. 2 (illustrating vulnerability database 212), 4.

55. After determining the vulnerabilities that apply to the software products installed on the host, Elder teaches that “[a] composite risk score can then be automatically determined for the host and each software product of the host.” *Id.*, ¶8 (describing aggregate risk score may also be employed); *see also id.*, ¶¶48. The risk scores take into account vulnerability scores associated with each identified vulnerability, which Elder teaches can be the base metrics of a CVSS score. *Id.*, ¶¶8, 44-48, 62-63, Figs. 2, 6; *see also id.*, ¶¶, 49, 64. Elder explains that the risk scores allow for the system 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 (invention “provide[s] automated enterprise risk analysis, quantification, and prioritization”), 50-51 (risk quantification includes attack graph generation that allows for the determination of “the least accessibility needed” of the vulnerabilities), 54, Fig. 8 (illustrating, e.g., “Vulnerability Risk Contribution Module 840”).

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

56. Kim is a publication of a patent application that relates to systems and methods for handling vulnerabilities associated with virtual machines in cloud environments. EX1006 (Kim), Abstract, ¶¶3 (describing “method and apparatus that automatically inspect the vulnerability of a virtual machine and recover damaged main system files in the hypervisor domain of a cloud computing environment”), 12, 41.

57. Kim teaches that the system collects “collection target information” from the virtual disk and virtual memory of each virtual machine using an API. *Id.*, Abstract, ¶¶13-15, 23-25, 40, 47-48, 50-51, Figs. 2 (illustrating virtual disk read/write API), 3 (illustrating virtual memory read API). The collection target information is used to inspect the virtual machine for vulnerabilities according to inspection criteria. *Id.*, ¶¶13, 53, 66-67, Fig. 4. Kim explains that the collection target information can include “main file information,” in addition to “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 (main file information collection module unit “collects main file information, such as the system files and registry of the virtual disk images”), 50 (virtual memory information collection module unit “collects the process execution information, kernel memory information, network port status information, etc. of the virtual machines”).

58. Inspection of the collected information for vulnerabilities is then performed, which can include, for example, “a software configuration and patch inspection” and “an operating system configuration and patch inspection.” *Id.*, ¶16; *see also id.*, ¶¶21 (“The vulnerability inspection criteria may be divided into software version management status, patch application status, operating system configuration status, network connection status, and process execution status, depending on attributes.”), 26, 44, 54.

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

59. Hufsmith is a publication of a patent application that describes security assessments involving the detection, prioritization, and reporting of security risks. EX1007 (Hufsmith), Abstract, ¶¶28-37. Hufsmith discusses assessing risks

associated with virtual resources (e.g., containers or VMs) that implement distributed applications. *Id.*, Abstract, ¶¶3-10, 149 (while containers are discussed in most embodiments, “it should be emphasized that the present techniques are applicable to other forms of encapsulated functionality, including virtual machine images”). Though sometimes referred to as a distributed application, singular, Hufsmith explains that a distributed application may include multiple applications performing various functions. *See, e.g., id.*, ¶¶3-4 (“The different components of such applications, such as those that expose functionality via a network address, can be characterized as services, which may be composed of a variety of other services, which may themselves be composed of other services. Examples of a service include an application component (e.g., one or more executing bodies of code) that communicates via a network (or loopback network address) with another application component, often by monitoring network socket of a port at a network address of the computer upon which the service executes.”).

60. Hufsmith teaches that its security assessment utilizes several scanners to detect various types of security risks, such as “a static analysis scanner; a dynamic analysis scanner; a malware analysis scanner; an antivirus scanner; or a configuration scanner.” *Id.*, ¶¶82, 228. These scanners are used to detect multiple types of vulnerabilities, including identifying installed applications that match

vulnerable applications listed in public repositories. *Id.*, ¶¶82 (scanners check “public repositories, such as repositories of CVE or CWE vulnerabilities”), 84 (same), 121 (obtaining information about security risks by querying “a repository, database, or other entity”), 160 (embodiments include “performing a lookup on existing vulnerability information in CVE and CWE databases”).

61. Hufsmith further teaches that each detected vulnerability may be evaluated and prioritized for overall risk. *Id.*, ¶¶6-7, 109. For example, a CVE evaluator may identify CVE properties contained within an asset, which represent “known potential CVEs or other known vulnerabilities and exposures” detected in the asset. *Id.*, ¶113, Fig. 1B. The CVE evaluator is configured to evaluate the information “to determine one or more CVE weights” that may be based on “metrics such as a vector describing the CVE,” such as a CVSS vector. *Id.*, ¶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 may include CVSS base metrics, such as “an attack vector 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”).

62. Hufsmith also teaches that the weights associated with the vulnerability may be further 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.” *Id.*, ¶¶92; *see also id.*, ¶¶94, 135-42. Contextual properties relating to the asset’s execution environment may indicate, for example, whether the asset or 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.” *Id.*, ¶140. “Depending on the information provided, metrics and values applicable to the execution environment may be selected and weights determined based on whether the execution environment increases or decreases the likelihood of metrics and values of a potential vulnerability.” *Id.*, ¶141. As an example, Hufsmith explains that if the execution environment of an asset 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.” *Id.* Contextual properties relating to historical information may include a history of risk scores determined from previous scans, which “may 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.*, ¶¶105-06, 139.

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

63. As I explain below, it is my opinion that claims 1-6, 8-9, and 11-18 were obvious over Elder and Kim. Elder describes systems and methods for automatically assessing the security of host devices on a network environment, such as a cloud computing environment. *Supra*, §XI.A (Elder summary). As part of this assessment, host configuration information of a host is accessed to determine the software applications installed on the host and their associated versions, which are then compared to a database of known software vulnerabilities to identify the vulnerabilities that are present on the host. *Id.* The vulnerabilities are then analyzed to quantify their risk by using CVSS score metrics for each vulnerability, which provide network accessibility information associated with the vulnerability. *Supra*, §§VIII.B.2, XI.A. Kim builds upon Elder by further discussing the benefits of accessing host configuration information using a cloud provider API and also teaching using network port status information of the host as part of the security assessment. *Supra*, §XI.B (Kim summary); *see also infra*, §XII.A (reasons to combine).

A. Reasons to Combine Elder and Kim

64. In my opinion, a POSA would have had reason to combine the teachings of Elder and Kim to arrive at claims 1-6, 8-9, and 11-18.

65. As an initial matter, Elder and Kim are analogous art to one another and to the '855 patent. Elder describes cyber security techniques involving analysis of host configurations, such as configurations of installed software applications, to identify vulnerabilities and assess risk to a cloud environment. *See supra*, §XI.A (Elder summary); EX1005 (Elder), Abstract, ¶¶5-8 (“[T]he present invention is implemented as a method for risk analysis,” which “includes accessing host configuration information of a host and querying a vulnerability database (e.g., the National Vulnerability Database (NVD)) based on the host configuration information”), 24 (“present systems and methods can be implemented as part of ... a cloud computing environment”), 28 (host information “includes configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each”), 30 (assessment “correlates the host configuration data against the vulnerability database data in order to quantify and prioritize the risks associated with deployed host configurations”). Kim similarly describes cyber security techniques that identify vulnerabilities based on the configurations of assets in a cloud computing environment. *See supra*, §XI.B (Kim summary); EX1006 (Kim), Abstract, ¶¶3 (“The present invention generally relates to an inspection and

recovery apparatus and method for handling virtual machine vulnerability ... in the hypervisor domain of a cloud computing environment.”), 13 (system collects “collection target information from each of virtual machines” and inspects “vulnerability of a given virtual machine ... based on the collected information”), 14-15, 22. The references are therefore in the same field of endeavor as the ’855 patent and are pertinent to issues addressed in the ’855 patent. *See, e.g.*, EX1001, 1:15-18 (“Disclosed embodiments include new systems and methods for securing cloud infrastructure”), 22:3-33 (“FIG. 5 of the disclosed embodiments describes a method of operating a cybersecurity system performing a side scanning function to protect against potential vulnerabilities.”).

66. Moreover, analyzing risk to a cloud environment by identifying software applications installed on a workload, determining their versions, and then comparing the data to a vulnerability database to identify vulnerabilities contained on the workload were well-known and predictable techniques. *Supra*, §VIII.B; EX1048 (Basavapatna), ¶¶37 (identifying “particular software products executing on the system”), 44, 53, 60, 79-80; EX1039 (Giakouminakis), 3:50-57; EX1020 (NIST Security Patches), 1-3 (known use of CVE matching); EX1041 (Milner), 2:24-67 (same); EX1042 (Khalid), 2:45-51, 8:1-30 (same); EX1044 (Czarny), 3:56-4:24 (hash-based matching), 11:27-47 (binary comparison); EX1059 (Hibbert),

¶¶51-51, 56. Further considering network accessibility information to assess the susceptibility of an attack from outside the workload was also a known technique. *Supra*, §VIII.B.2; EX1049 (CVSS Version 3), 3; EX1048 (Basavapatna), ¶¶15, 22, 55-56, 59; EX1039 (Giakouminakis), 4:36-50, 5:4-22, 8:46-9:3; EX1045 (Mohanty), 3:43-54, 4:41-46, 5:16-6:1, 8:47-50; EX1007 (Hufsmith), ¶¶92, 116-17, 140-41; EX1059 (Hibbert), ¶¶76-81. The applied teachings of Kim—including using a cloud provider API to access workload configuration information and using at least one port as part of the security assessment—were similarly well-known and predictable. *Supra*, §§VIII.A, VIII.B.2; EX1048 (Basavapatna), ¶¶22, 37 (port settings); EX1051 (Nyanchama), ¶¶5-8 (same); EX1052 (Cole), ¶6 (same); EX1057 (Murthy), ¶¶39, 57; EX1058 (Murchison), EX1018 (Price), ¶21 (use of API to access configuration information). Thus, in my opinion, the combination of Elder and Kim merely involves using these well-known techniques to perform their known functions with predictable results.

67. A POSA would also have been motivated to combine Elder and Kim for several reasons. At the outset, I note that Kim expressly identifies Elder as a “related preceding technology” that discloses an “automation system, which collects the configuration information of a host, 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 (Kim), ¶10. As I explain more below, a POSA would have understood Kim’s teachings as improving upon a vulnerability risk assessment like that taught by Elder through the use of cloud provider APIs to gather host configuration information and the consideration of port information for vulnerability analysis.

68. First, a POSA would have been motivated to combine the teachings of Elder and Kim to use a cloud provider API to access host configuration information stored in a block storage volume of a workload. For example, Kim acknowledges that systems for performing automated vulnerability inspections of cloud resources, such as virtual machines, were known in the art. EX1006 (Kim), ¶¶5-11 (“Such a host vulnerability automatic inspection agent is advantageous in that inspections, such as the maintenance of the latest security patches of an OS and applications and the configuration of major security, may be automated.”). Kim explains that these systems often used an agent installed in each virtual machine, but such agent-based techniques raised risks “that the results of inspection may be forged by a malicious user or malicious code.” *Id.*, ¶9. s Elder describes one such system that uses an installed agent to gather host configuration information, such as software applications installed on the host, for security assessments. EX1005 (Elder), ¶¶33 (“Host 202 includes agent 216 which automatically collects and accesses host

configuration datastore 220 from host 202.”), 34 (explaining “agent 216 may collect configuration information from the registry of a system with the Windows® operating system”), 77, Fig. 2; *see also* EX1006 (Kim), ¶10 (referencing Elder).

69. Kim improves upon agent-based techniques by using an agentless method to gather host configuration information to inspect for vulnerabilities and does so by utilizing APIs provided in the cloud environment to access such information. EX1006 (Kim), ¶¶8-9, 12-15. Indeed, as I described above, the benefits of agentless scanning methods were already well-known and appreciated in the art. *Supra*, §VIII.B.1.

70. In my opinion, a POSA would have been motivated to use a cloud provider API, such as taught by Kim, to obtain host configuration information for vulnerability assessments, such as taught by Elder. Elder explains host configuration information, including applications installed on the host, can be obtained from files stored on the host, such as “the registry of a system with the Windows® operating system.” EX1005 (Elder), ¶34 (“The installed programs can be collected from the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall key which includes high level data regarding the operating system, applications that are installed, operating system hotfixes/updates, and application patches.”); *see also id.* (“Configuration data may further be collected based on an

enumeration of the portable executable (PE) files (e.g., executable (.exe) files and dynamic link library (DLL) files) on the file system.”). Kim similarly teaches collecting configuration information from the same types of stored files (e.g., registry files, application configuration files, process execution information) by using a cloud provider API. EX1006 (Kim), ¶¶14 (“Collecting the collection target information may include collecting main file information of a virtual disk image of each virtual machine via a virtual disk read/write Application Programming Interface (API).”), 15 (collection target information may also “be obtained from virtual memory of each virtual machine via a virtual memory read API and that includes process execution information, kernel memory information, and network port status information of the virtual machine”), 44 (“[T]he collection target information may be registry and OS main configuration files, application configuration files, network connection information, process execution information, etc.”). Thus, a POSA would have appreciated that accessing the same type of information needed to perform Elder’s vulnerability assessment could readily be performed using a cloud provider API, thereby gaining the benefits of an agentless technique taught by Kim and known in the art.

71. Second, a POSA would have been motivated to combine the teachings of Elder and Kim to more accurately assess the risk of attack from outside the

workload. As recognized by Elder, “software applications and operating systems have become increasingly complex and the number of security vulnerabilities has increased.” EX1005 (Elder), ¶4. Thus, “analysis of security vulnerabilities has become increasingly important as security vulnerabilities may expose an enterprise to significant harm, including theft of valuable and sensitive information, damage to valuable work products (e.g., deletion or corruption of files, data, or software programs, etc.), and system damage causing loss of productivity, revenue, etc.” *Id.* To identify risky software applications and operating systems, Elder describes the well-known technique of matching software applications that are installed on a host to vulnerabilities known to be associated with such applications. EX1005 (Elder), ¶¶7-10, 28-30; *see also* EX1006 (Kim), ¶10; *supra*, §VIII.B.2. In addition to using this known matching technique, Elder teaches that the risk assessment can be further improved by identifying those vulnerabilities that present the most risk to exposing an enterprise to harm. EX1005 (Elder), ¶¶7-8, 30, 54. Elder specifically teaches determining a CVSS score for each identified software vulnerability, which provides metrics describing the type of network accessibility parameters needed to exploit the vulnerability and thus gain access to the host, and using those metrics to assess the vulnerability’s risk of attack. *Id.*, ¶¶8, 43-49; *see also supra*, §VIII.B.2.

72. Kim also teaches the desirability to provide systems that automatically perform security checks for vulnerabilities present on cloud computing resources, such as virtual machines. EX1006 (Kim), ¶¶5-7 (noting traditional use of antivirus programs and firewalls “does not desirably perform significant security check tasks that are to be performed after being installed, for example, the maintenance of latest security patches for software itself or other applications and the configuration and management of operating system (OS) security”). Kim recognizes that matching installed software applications to a vulnerability database to identify vulnerabilities present on the host and computing overall risk of those vulnerabilities was a well-known technique, specifically mentioning Elder as one such system. EX1006 (Kim), ¶10 (stating Elder “collects the configuration information of the host, analyzes the collected information based on information stored in a vulnerability database (DB), and calculates vulnerability scores”).

73. Kim also expressly teaches gathering additional network-related information, such as the status of network ports on the host, as part of a vulnerability assessment, which a POSA would have recognized would further inform the assessment of risk that a vulnerability could be exploited by an attacker outside the workload. *Id.*, ¶¶15, 22, 50. As I explained above, POSAs understood that the status of network ports (e.g., whether a given port was open or closed) was useful in

assessing whether a vulnerable application would be capable of receiving data from outside the host, which increases the risk that the vulnerable application could be exploited by a malicious attacker. *Supra*, §VIII.B.2; EX1051 (Nyanchama), ¶¶5-8 (applications commonly use data ports to facilitate communication); EX1052 (Cole), ¶6 (open ports provide entry points for possible intrusion by a malicious hacker); EX1058 (Murchison), 4:10-36 (attackers seek out open ports to target vulnerable applications); EX1057 (Murthy), ¶¶39, 57 (mitigating risk of attack by closing port access to vulnerable software); EX1048 (Basavapatna), ¶22 (same). Thus, a POSA would have sought to further use additional network-related information, such as at least one port, as part of the risk assessment, as taught by Kim, in determining the risk that a given software vulnerability presents to the computing environment, such as taught by Elder, because doing so would have additionally informed how the vulnerability may be susceptible to attack from outside the workload.

74. Finally, a POSA would have had a reasonable expectation of success in combining the teachings of Elder and Kim to arrive at the claimed inventions. As I have explained above, the software-matching and vulnerability risk assessment techniques disclosed in Elder were well understood and routinely practiced. *Supra*, §VIII.B.2; EX1048 (Basavapatna), ¶¶37, 44, 53, 60, 79-80; EX1041 (Milner), 2:24-67; EX1042 (Khalid), 2:45-51, 8:1-30; EX1044 (Czarny), 3:56-4:24, 11:27-47;

EX1059 (Hibbert), ¶¶41-51, 56. Further, gathering the configuration data necessary to perform such analyses, including port information, using a cloud provider API, as taught by Kim, was similarly well-known and predictably implemented. *Supra*, §VIII.A; EX1018 (Price), ¶21. A POSA would have been familiar with the techniques applied in the combination of Elder and Kim—including, for example, identifying software applications installed on the workload, determining their associated versions, comparing the data to a data structure of known vulnerabilities to identify vulnerabilities associated with the workload, and determining the identified vulnerabilities that are susceptible to outside attack based on network accessibility information and at least one port—and combining the references’ teachings would not have presented meaningful technical challenges or produced unexpected results. *Supra*, §VIII.B.2.

B. Independent Claims 1, 12, and 18

75. Claims 1, 12, and 18 recite a system, method, and computer-readable medium, respectively, while reciting analogous limitations, which are discussed together below.

1. Preambles

76. As I explain below, Elder, alone or in combination with Kim, teaches the preambles of claims 1, 12, and 18. Claim 1 recites a “cyber security system for a cloud environment, the system comprising.” Claim 12 recites a “method,

comprising.” The preamble of 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.”

77. Elder describes security methods and systems for automated risk analysis of computer systems based on host configuration data and vulnerability information. EX1005 (Elder), Abstract, ¶¶3 (“Embodiments of the present invention are generally related to digital computer systems and computer system security.”), 6-8, 31, claims 1, 17. Elder further teaches that the described methods and systems “can be implemented in a variety of architectures and configurations,” including, for example, in “a cloud computing environment.” *Id.*, ¶24; *see also id.*, ¶27 (“It is appreciated that operating environment 100 may be part of ... a cloud computing environment....”), Fig. 1. Similarly, Kim teaches cyber security methods and systems for performing vulnerability inspections in a cloud environment. EX1006 (Kim), Abstract (describing “[a]n inspection and recovery method and apparatus” for “providing a secure virtual machine use environment for cloud computing”), ¶¶3 (method and apparatus “automatically inspect the vulnerability of a virtual machine and recover damaged main system files in the hypervisor domain of a cloud computing environment”), 7, 12, 41, Fig. 1. Thus, Elder and Kim teach a “cyber

security system for a cloud environment” and corresponding methods, as recited by claims 1 and 12.

78. With respect to the “non-transitory computer readable medium” of claim 18, Elder teaches its systems include a computer-readable medium that is non-transitory (e.g., hard disks and random-access memory) and stores instructions that cause a computing device to perform the system’s operations. EX1005 (Elder), ¶¶24 (“Embodiments described herein may be discussed in the general context of computer-executable instructions residing on some form of computer-readable storage medium, such as program modules, executed by one or more computers, computing devices, or other devices. By way of example, and not limitation, computer-readable storage media may comprise computer storage media and communication media.”), 25 (“Computer storage media can include, but is not limited to, random access memory (RAM), read only memory (ROM), electrically erasable programmable ROM (EEPROM), flash memory, or other memory technology....”). Elder also teaches that a single system can perform the operations associated with the automated risk analysis. *Id.*, ¶¶9 (“In one embodiment, the present invention is implemented as a computer readable storage medium having stored thereon, computer executable instructions that, if executed by a computer system, cause the computer system to perform a method including”), 10 (the

computer system may have “one or more processors coupled to a computer readable storage media and executing computer readable code which causes the computer system to implement” the described methods), 24 (“The functionality of the program modules may be combined or distributed as desired in various embodiments.”), 55, 71-72, 80-81, Figs. 8-9, claim 9. In the context of the combination of references that I discuss in this ground and the combinations that I discuss in subsequent grounds, a POSA therefore would have understood that execution of the instructions contained on the medium would “cause the at least one processor to perform” each of the operations recited in claim 18.

79. It is therefore my opinion that Elder, alone or in combination with Kim, teaches the preambles of claims 1, 12, and 18.

2. Element 1.i

80. Claim 1 recites “at least one processor configured to....” As I explain below, Elder teaches this limitation.

81. Elder’s systems that I discussed in the previous section include at least one processor. For example, Elder explains the systems may be “implemented as a clustered computing system comprising a computer system having one or more processors” EX1005 (Elder), ¶10; *see also id.*, ¶¶55 (described processes, “in various embodiments, are carried out by a processor”), 71, 80-81, Figs. 8-9. The processors are configured to execute instructions to perform the system’s operations.

Id., ¶¶10, 55. As I explained above, Elder also teaches that a single system can perform the operations associated with the automated risk analysis. *Id.*, ¶¶9-10, 24, 55, 71-72, 80-81, Fig. 8-9, claim 9; *supra*, §XII.B.1. In the context of the combination of references that I discuss in this ground and the combinations that I discuss in subsequent grounds, a POSA therefore would have understood the system's operations to include each of the operations recited in claim 1 and its dependent claims.

82. It is therefore my opinion that Elder teaches element 1.i.

3. Elements 1.1, 12.1, and 18.1

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

84. First, Elder teaches accessing a block storage volume of a workload maintained in a cloud storage environment. For example, Elder teaches that its systems and methods may perform risk analysis on hosts 102a-c (i.e., workloads) contained within an operating environment 100, which “may be a variety of computing systems includ[ing] desktop computers, point of sale systems, laptops, netbooks, phones, smart phones, etc.” EX1005 (Elder), ¶27, Fig. 1; *compare* EX1001, 16:54-56 (“Workloads, as used herein, may refer to systems, devices, or resources in a network (or available via a network) such as a cloud infrastructure.”).

As I explained above, Elder teaches that the operating environment 100 may be a cloud computing environment. EX1005 (Elder), ¶¶24, 27; *supra*, §XII.B.1. Thus, a POSA would have understood that the hosts 102a-c contained within the cloud computing environment would also include resources common to a cloud environment, such as virtual machines and containers. *Supra*, §VIII.A (cloud computing overview).

85. The system “collects configuration data from hosts 102a-c for risk analysis,” which includes “details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each.” EX1005 (Elder), ¶28. To collect such data, Elder teaches that “[h]ost 202 includes agent 216 which automatically collects and accesses host configuration datastore 220 from host 202.” *Id.*, ¶33; *see also id.*, ¶¶35-36, 57 (“At block 702, host configuration information of a host is accessed,” which “comprises information about a plurality of software applications and an operating system of a host”), 77, Figs. 3, 7, claims 1, 9, 17. For example, host configuration data may be obtained by accessing “the registry of a system with the Windows® operating system, available from Microsoft Corporation of Redmond, Wash.,” or other stored files such as “portable executable (PE) files (e.g., executable (.exe) files and dynamic link library (DLL) files) on the file system.” *Id.*, ¶34; *compare* EX1001, 18:32-34 (“Accessing block storage volumes,

as used herein, may refer to obtaining, examining, or retrieving data.”). These types of files were commonly understood to be data files maintained in block storage volumes, such as, for example, virtual disks of virtual machines. See EX1005 (Elder), ¶81 (“Applications resident with computer system 910 are generally stored on and accessed via a computer readable medium, such as a hard disk drive (e.g., fixed disk 944), an optical drive (e.g., optical drive 940), a floppy disk unit 937, or other storage medium.”); EX1006 (Kim), ¶¶14, 22, 47 (configuration information files, such as registry files, accessed on the virtual disk of a virtual machine); EX1053 (Veselov), 11:5-33 (virtual disk drive of virtual machine stores “data implementing and/or used by the virtual machine 312; such data may include an operating system 360, applications 364 (i.e., software packages) that can be executed by the operating system 360, and other files 366 which in various embodiments may be organized and/or accessed using a file system (e.g., NTFS, procfs)”). Because Elder teaches that the operating environment 100 may be cloud environment (*id.*, ¶¶24, 27; *supra*, §XII.B.1), Elder teaches accessing a block storage volume of a workload maintained in a cloud storage environment (e.g., files stored on a cloud-based host, such as on a virtual disk of a virtual machine), as recited. Compare EX1001, 18:26-36 (“Block storage volumes, as used herein, may refer to data files

maintained ... in cloud-based storage environments.”), 39:5-7 (“[A] block storage volume may be implemented as a virtual disk.”), 39:10-17.

86. While Elder teaches that its risk analysis systems and methods may be for a cloud computing environment, Elder does not explicitly disclose using a cloud provider API to access the block storage volume of the workload. *Supra*, §XII.A. However, as I explained above, a POSA would have sought to use a cloud provider API, rather than an agent installed on the host, to access and obtain the host configuration information in view of Kim. *Id.*

87. More specifically, Kim teaches systems and methods “for handling virtual machine vulnerability” in a cloud environment. EX1006 (Kim), Abstract; *see also id.*, ¶¶3 (“The present invention generally relates to ... a method and apparatus that automatically inspect the vulnerability of a virtual machine ... of a cloud computing environment.”), 12, 40, Fig. 1. To identify vulnerabilities present in workloads (e.g., virtual machines), Kim teaches collecting “collection target information” from each of the virtual machines using an API provided by the cloud environment. *Id.*, ¶¶13-14 (teaching use of “a virtual disk read/write Application Programming Interface (API)”); *see also id.*, ¶¶15 (further teaching “[c]ollecting the collection target information may include collecting information that can be obtained from virtual memory of each virtual machine via a virtual memory read API”), 24-

25, 40, 47-48, 50-51, Figs. 1-4, claims 1-3, 11-13. Similar to Elder, Kim teaches that “[t]he 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. A POSA would have understood the APIs taught by Kim to access stored configuration files and related information for vulnerability analysis are cloud provider APIs because it is the provider of the cloud environment that provides access to such files. *See supra*, §VIII.A (use of APIs in cloud computing); EX1018 (Price), ¶21 (“Virtual machine manager 130 interfaces can include interfaces and application programming interfaces (APIs) that can provide operations and accessing including guest management, offline registry access, virtual disk access, and other features of virtual machines that maybe running or accessible through a particular virtualization host environment.”). Thus, Kim teaches using a cloud provider API to access a block storage volume of a workload (e.g., a virtual machine) maintained in a cloud storage environment, as recited. *Compare* 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. A cloud provider[] may include any entity through whom information is accessible over the Internet or other shared network.”). A POSA therefore 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 to obtain host configuration information for risk analysis, as taught by Elder. *Supra*, §XII.A.

88. It is therefore my opinion that the combination of Elder and Kim teaches all aspects of elements 1.1, 12.1, and 18.1.

4. Elements 1.2, 12.2, and 18.2

89. These elements recite “[identify/identifying] an installed software application in the accessed block storage volume.” As I explain below, Elder teaches these elements.

90. Elder teaches identifying software applications that are installed on the host devices from the accessed block storage volume. Specifically, Elder teaches that the accessed host configuration information “comprises information about a plurality of software applications and an operating system on the host.” EX1005 (Elder), ¶7; *see also id.*, ¶¶28 (the data collected “includes configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each”), 34, 57, Figs. 3, 7, claims 1-2. As I explain in more detail below, Elder compares this information to a database containing known vulnerabilities to identify vulnerabilities that are present on the host. *Id.*, ¶¶37-39, Fig. 4; *infra*, §§XII.B.6-7.

91. To make this comparison, a POSA would have understood that Elder's system identifies the software applications that are installed on the host as indicated by the accessed host configuration information. As one example, Elder explains that "[t]he installed programs can be collected from the HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Uninstall key which includes high level data regarding the operating system, applications that are installed, operating system hotfixes/updates, and application patches." EX1005 (Elder), ¶34. A POSA understood that, by using this registry key, a list of subkeys that correspond to applications installed on the host is provided. See, e.g., EX1054 (Windows Installed Software), 4-5 (describing use of Microsoft Sysinternals programs for "interrogat[ing] the *HKLM/System* registry key to get the listing of installed programs by using the Windows remote registry service"), 8 (describing use of Microsoft PowerShell command shell to open registry key location and read its contents).²⁶ As another example, Elder teaches that

²⁶ EX1054 is a copy of "Windows Installed Software Inventory: Gathering the Information Needed For the 20 Critical Controls," obtained from

“[c]onfiguration data may further be collected based on an enumeration of the portable executable (PE) files (e.g., executable (.exe) files and dynamic link library (DLL) files) on the file system,” which “provides a comprehensive listing of each software program that could be run on a host, independent of whether [the] program was installed properly within the operating system, downloaded (e.g., intentionally or otherwise), or remaining as a result of an improper uninstall.” EX1005 (Elder), ¶34. This data thus allows the system to identify an installed software application in the accessed block storage volume, as recited.

92. It is therefore my opinion that the combination of Elder and Kim teaches all aspects of elements 1.2, 12.2, and 18.2.

5. Elements 1.3, 12.3, and 18.3

93. These elements recite “[analyze/analyzing] the identified installed software application to determine an associated software version.” As I explain below, Elder teaches these elements.

<https://www.giac.org/paper/gccc/554/windows-installed-software-inventory/121403>.

94. Elder teaches the installed software applications as identified by the host configuration information are further analyzed to determine their associated software versions. As I explained above, Elder teaches accessing stored host files to identify software applications installed on the host, which are used to compare to a database of known vulnerabilities. *Supra*, §XII.B.4. Elder explains that the stored data can also include information of a software version associated with each installed application. EX1005 (Elder), ¶¶28 (“[T]he data collected by risk analysis module 108 includes configuration details regarding hardware, operating system, patches, hotfixes, applications, and associated versions of each.”), 34.

95. Elder further teaches that, as part of identifying vulnerabilities associated with the installed software applications, the system “issues program name query 420 to vulnerability database 412 which includes the list of programs stored in program datastore 432.” *Id.*, ¶37, Fig. 4; *see also id.*, ¶35 (explaining collected host configuration information is stored by system in the program datastore). Elder describes that “the most precise version number available for each software program” is used for the query and “may include program names based on a plurality of common platform enumerations (CPEs).” *Id.*, ¶¶37-38.

96. The Common Platform Enumeration (CPE) was a well-known and routinely used standard for enumerating attributes associated with software

applications present on a given host. EX1055 (CPE Specification), 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.”). The CPE standard was known to provide a logical construction in which a system could set values for a defined set of attributes in order to uniquely identify a given software application. *Id.*, 1 (“IT management tools can collect information about installed products, identify products using their CPE names, and use this standardized information to help make fully or partially automated decisions regarding the assets,” including “enforcing and verifying IT management policies relating to these assets, such as vulnerability, configuration, and remediation policies.”), 8-9 (explaining a “well-formed CPE name (WFN)” can “be used to describe a set of products or to identify an individual product”).

97. The set of attributes defining a CPE name included attributes relating to 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, among others, and values for each attribute could be set to define a specific product (or group of products). *Id.*, 9, 12. For example, for “Microsoft Internet Explorer 8.0.6001 Beta (no edition),” a well-formed CPE identifier would be defined as:

```
wfn:[part="a",vendor="microsoft",product="internet_explorer",  
version="8\.0\.6001",update="beta",edition=NA]
```

where the value “a” indicated that the product was part of the applications class, Microsoft was the vendor, the product’s name was Internet Explorer, and its version was 8.0.6001. *Id.*, 14.

98. Known vulnerabilities listed in public repositories, such as the National Vulnerability Database (NVD), were routinely linked to a CPE identifier associated with the software application(s) affected by the vulnerability. EX1005 (Elder), ¶¶39 (teaching known vulnerabilities include “common vulnerability enumerations (CVEs)”), 45 (explaining “each CVE vulnerability enumerates vulnerable configuration elements with a Common Platform Enumeration (CPE) identifier”); EX1031 (NIST NVD Overview), 2 (explaining 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”). Thus, a POSA understood that, by enumerating the attributes of a software application installed on the host according to the CPE standard, the software application could be more accurately matched to any vulnerabilities known to be associated with that particular application. *See* EX1005 (Elder), ¶¶38-39 (explaining “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 cross-list the vulnerability database “to map

vulnerabilities to only those exact products and versions to which the vulnerabilities apply”).

99. Because Elder teaches enumerating the installed software applications identified from the stored host files as CPEs, a POSA would have understood that Elder’s system analyzes the identified installed software applications to determine associated software versions. For example, as I detailed above, Elder teaches that software applications installed on a host having the Windows operating system may be identified using the Uninstall registry key, which provides a listing of subkeys that correspond to installed applications. EX1005 (Elder), ¶34; *supra*, §XII.B.4. The subkeys themselves each could be accessed to obtain more specific data relating to the corresponding installed application, such as the application’s version number. See EX1054 (Windows Installed Software), 8-9, Figs. 8-9. In this manner, the identified installed application would be analyzed so that the version number could be retrieved (by, for example, extracting the data listed for the “DisplayVersion” name of the subkey). See *id.*, 8-9, 22-27 (outlining PowerShell script for extracting version value provided under “DisplayVersion” property of application subkey); EX1005 (Elder), ¶37 (“Embodiments of the present invention use the most precise version number available for each software program.”). The retrieved data could then be set as the value for the version attribute of the application’s CPE. See

EX1055 (CPE Specification), 12 (values for version attribute should be “copied directly ... from discoverable data”), 14 (describing set functions for adding attribute-value pairs to WFN). A POSA would have understood that a similar process would occur for other stored host files Elder teaches accessing to identify installed software applications (e.g., executable or library files). See EX1005 (Elder), ¶34. Thus, Elder teaches analyzing the identified installed application to determine an associated software version, as recited. Compare EX1001, 23:41-60 (system “may include functionality to analyze installed software applications to determine the associated software version,” which “may identify the software version based on unique version name, unique version number, and may be based on unique states of the currently installed computer software”), 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”).

100. It is therefore my opinion that the combination of Elder and Kim teaches all aspects of elements 1.3, 12.3, and 18.3.

6. Elements 1.4, 12.4, and 18.4

101. These elements recite “[access/accessing] a data structure of known software vulnerabilities for a plurality of versions of software applications.” As I explain below, Elder teaches these elements.

102. To perform its risk assessment, Elder’s system includes a “vulnerability database” (i.e., a data structure) that can be accessed to identify vulnerabilities associated with the identified software applications installed on the host. EX1005 (Elder), ¶¶7, 29 (system “maintains an updated database of vulnerabilities (e.g., such as that provided by the National Vulnerability Database (NVD), fix databases, etc.)”), 37-39 (risk analysis module 408 queries vulnerability database 412 to retrieve list of vulnerabilities that apply to the host), Figs. 2, 4, 7. Elder teaches that the vulnerability database includes data on known software vulnerabilities, such as vulnerabilities from the National Vulnerability Database (NVD). *Id.*, ¶¶7, 31-32 (“Vulnerability database 212 contains information including security vulnerabilities associated with installed software programs.”), 38; *supra*, §VII.B.1 (describing known public repositories of software vulnerabilities). Each of these known vulnerabilities are associated with a particular version of a given software application and thus represent known software vulnerabilities for a plurality of versions of software applications, as recited. EX1005 (Elder), ¶¶29 (“The database catalogs vulnerabilities associated with configuration elements including operating systems, applications, and particular versions of each that are vulnerable.”), 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.”), 39 (“In one

embodiment, list of vulnerabilities 426 includes a plurality of common vulnerability enumerations (CVEs.”); *compare* EX1001, 14:1-15 (vulnerability database may include vulnerability data from NVD).

103. It is therefore my opinion that Elder teaches all aspects of elements 1.4, 12.4, and 18.4.

7. Elements 1.5 and 12.5

104. These elements recite “[perform/performing] a lookup of the identified installed software version in the data structure to identify known vulnerabilities.” As I explain below, Elder teaches these elements.

105. Elder teaches that its system is configured to perform a lookup of the identified software version in the vulnerability database to identify which of the known vulnerabilities stored in the database apply to the software applications installed on the host. For example, Elder describes that, after identifying the software applications installed on the host and their associated versions, the data is stored in a program datastore of the risk analysis module. EX1005 (Elder), ¶¶35-36 (program datastore 332 stores “a list of software programs installed on a plurality of hosts”), Fig. 4. To identify vulnerabilities that are associated with the installed applications, the “[r]isk analysis module 408 issues program name query 420 to vulnerability database 412 which includes the list of programs stored in program datastore 332.” EX1005 (Elder), ¶37. The program name query 420 issued by the

risk analysis module 408 identifies the installed program names and associated versions by, for example, their corresponding CPEs. *Id.*, ¶¶37-38; *see also supra*, §XII.B.5. “Vulnerability database 412 responds with program name results 422 which includes program names and versions that have vulnerabilities.” EX1005 (Elder), ¶37.

106. Elder further teaches that the risk analysis module 408 is then configured to issue a program name query 424, which includes a list of programs identified as having vulnerabilities by program name results 422, to the vulnerability database. *Id.*, ¶39. “Vulnerability database 412 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.*; *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.”), 39 (“In one embodiment, list of vulnerabilities 426 includes a plurality of common vulnerability enumerations (CVEs).”), 58-59, Fig. 7.

107. It is therefore my opinion the combination of Elder and Kim teaches all aspects of elements 1.5 and 12.5.

8. Elements 1.6, 12.6, and 18.5

108. These elements 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.” As I explain below, the combination of Elder and Kim teaches these elements.

109. As an initial matter, I note that, like elements 1.6 and 12.6, element 18.5 recites using network accessibility information and at least one port “to identify one or more of the known and identified vulnerabilities” that are susceptible to attack from outside the workload. However, unlike claims 1 and 12, claim 18 does not recite performing a lookup “to identify known vulnerabilities” (i.e., claim elements 1.5 and 12.5, discussed above). I understand that, during prosecution of the ’972 application, the pending claim that would ultimately issue as claim 18 of the ’855 patent originally recited the same lookup element as claims 1 and 12 (i.e., elements 1.5 and 12.5). EX1004, 1237-38 (claim 40). I also understand that, while this element did not appear in later claim listings, there appears to be no indication from the applicant that this element was intentionally removed from the claim. *See, e.g.*, EX1004, 774 (claim 40). Accordingly, for the purposes of my analysis below, I treat “the known and identified vulnerabilities” recited in element 18.5 as the identified

vulnerabilities that I discussed above with respect to elements 1.5 and 12.5. *Supra*, §XII.B.7.

110. 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. For example, after identifying which of the known vulnerabilities apply to the specific versions of software applications installed on the host, Elder explains that the system may be further configured to determine a risk score for each of the software products. EX1005 (Elder), Abstract, ¶¶8, 46 (“Risk analysis module 608 automatically provides detailed risk analysis, quantification, and prioritization based on each configuration element (e.g., programs and the operating system) installed on each host and a risk score is calculated”), 54, 62-64, Fig. 7. Elder teaches that this determination involves retrieving a vulnerability score for each of the known software vulnerabilities that were identified as applying to the host. *Id.*, Abstract, ¶¶7-8, 32, 43-44 (“Vulnerability score database 612 returns vulnerability scores 622 which are used by risk analysis module 608 to determine risk tuples for each software program and client or host.”), 62, Figs. 6-7. The vulnerability score may include “Common Vulnerability Scoring System (CVSS) scores.” *Id.*, ¶44; *see also*

id., ¶¶7, 45-46 (“CVSS scores are assigned to each vulnerability with a Common Vulnerability Enumeration (CVE) identifier.”).

111. The CVSS score information used by Elder’s system would have been understood to provide network accessibility information that includes data from an external data source, as recited. As Elder explains, “CVSS is part of Security Content Automation Protocol (SCAP) set of standards, developed by the U.S. Government, National Institute of Standards and Technology (NIST), MITRE, and others” and “provides a universal open and standardized method for rating IT vulnerabilities.” *Id.*, ¶45. Elder teaches that, “[i]n one embodiment, the base metrics of CVSS version 2 are used for risk quantification calculations,” which “measure the fundamental qualities of a vulnerability.” *Id.*, ¶47. As Elder explains, the base metrics of CVSS version 2 include data relating to “three different accessibility characteristics (e.g., access vector, access complexity, and authentication)” for a given vulnerability. *Id.* 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. See EX1031 (NIST NVD Overview), 2 (CVSS is “a set of metrics for communicating the characteristics and impacts of IT vulnerabilities”).

112. A POSA would have understood that each of these base metric characteristics provided “network accessibility information” as recited in the claims

because the characteristics conveyed information regarding the network access conditions needed for an outside attacker to exploit a given vulnerability on the host. For example, it was understood that the access vector, access complexity, and authentication metrics of CVSS version 2 “capture how the vulnerability is accessed and whether or not extra conditions are required to exploit it.” EX1050 (CVSS Version 2), 7. The access vector “reflects how the vulnerability is exploited” and indicated the type of network access that was needed for an attacker to exploit a given vulnerability. *Id.* For instance, a vulnerability with an access vector value of “Local (L)” meant that an attacker needed local access (such as “physical access to the vulnerable system or a local (shell) account”) to exploit the vulnerability, while a vulnerability with an access vector value of “Adjacent Network (A)” required an outside attacker “to have access to either the broadcast or collision domain of the vulnerable software,” such as access through local networks including “local IP subnet, Bluetooth, IEEE 802.11, and local Ethernet segment.” *Id.* A vulnerability with an access vector value of “Network (N),” in contrast, indicated that the vulnerable software was “bound to the network stack” and “does not require local network access or local access,” meaning the vulnerability was remotely exploitable. *Id.* “The more remote an attacker can be to attack a host, the greater the vulnerability

score” (i.e., the more likely the vulnerability would be susceptible to attack from outside the workload). *Id.*

113. The access complexity and authentication metrics also provided network accessibility information that was relevant to a vulnerability’s susceptibility to outside attack. The access complexity metric “measures the complexity of the attack required to exploit the vulnerability once an attacker has gained access to the target system,” such as whether the attacking party needed elevated privileges to exploit the vulnerability (indicating a “High (H)” access complexity) or whether the affected product “typically requires access to a wide range of systems and users, possibly anonymous and untrusted (e.g., Internet-facing web or mail server)” (indicating a “Low (L)” access complexity). *Id.*, 8 (“The lower the required complexity, the higher the vulnerability score.”). Further, the authentication metric “measures the number of times an attacker must authenticate to a target in order to exploit a vulnerability.” *Id.*, 8. For example, an authentication metric value of “None (N)” indicates that “[a]uthentication is not required to access and exploit the vulnerability.” *Id.*, 9; *see also id.*, 8 (“The fewer authentication instances that are required, the higher the vulnerability score.”).

114. I note that by April 26, 2021, a version 3 of the CVSS framework had issued. *See* EX1049 (CVSS Version 3); *supra*, §VIII.B.2 (providing overview of

CVSS scores). The standard for CVSS version 3 also used base metrics, which included attack vector, attack complexity, and privileges metrics. *Id.*, 5-8 (describing exploitability metrics). These metrics largely corresponded to the accessibility metrics of CVSS version 2. For example, like the access vector of version 2, the attack vector of version 3 reflected “the context by which vulnerability exploitation is possible.” *Id.*, 6. It included metric values of “Network (N)” (indicating “the vulnerable component is bound to the network stack and the attacker’s path is through open systems interconnection (OSI) layer 3 (the network layer)”), “Adjacent (A)” (indicating the vulnerable component was bound to the network stack, but limited to the same shared physical or logical network), “Local (L)” (indicating the vulnerable component was not bound to the network stack and “the attacker’s path is via read/write/execute capabilities”), and “Physical (P)” (indicating the attacker must “physically touch or manipulate the vulnerable component”). *Id.*; *see also id.*, 6-7 (describing attack complexity metric as indicating the types of network conditions needed to exploit the vulnerability), 7 (describing privileges metric as the type of network authorization needed to carry out an attack); EX1007 (Hufsmith), ¶117 (describing same).

115. Thus, a POSA would have understood that the CVSS score metrics would include network accessibility information whether using version 2 (as

expressly disclosed by Elder) or version 3 as made available by the relevant timeframe.

116. Elder also teaches using the network accessibility information provided by the base metric characteristics of the CVSS as part of its risk analysis. Specifically, POSAs understood 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 that I described above. EX1050 (CVSS Version 2), 1; *see also id.*, 2 (“With CVSS, anyone can see the individual characteristics used to derive score.”), 5 (stating “the vector should always be displayed with the vulnerability score”); EX1049 (CVSS Version 3), 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”).

117. To illustrate this, one example is a vulnerability identified by CVE-2003-0818, which relates to a vulnerability associated with the Microsoft Windows ASN.1 library that, if successfully exploited, could result in a buffer overflow condition that would allow an attacker to execute arbitrary code with administrative privileges. EX1050 (CVSS Version 2), 20-21. The CVSS base score for this

vulnerability is reported at a maximum of 10.0 and includes a vector string showing the base metrics used to derive the score as follows:

AV:N/AC:L/Au:N/C:C/I:C/A:C.

where the access vector (AV) value of “N” shows that the vulnerability is remotely exploitable, the access complexity (AC) value of “L” shows that low complexity is required to exploit the vulnerability, and the authentication (Au) value of “N” shows that no authentication is required. *Id.*, 20; *see also* EX1049 (CVSS Version 3), 23 (illustrating similar vector representation).

118. 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 (Elder), ¶¶44, 47. For example, as I explain 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.

119. A POSA would have understood that Elder’s use of network accessibility information provided by the CVSS score base metrics enabled Elder’s system to identify one or more of the known and identified vulnerabilities

susceptible to attack from outside the workload, as recited. As I described above, each of the base metrics of the CVSS framework informs the network access conditions needed for a vulnerability to be exploited by an outside attacker and thus measures the degree to which a given vulnerability is susceptible to attack from outside the workload. Furthermore, Elder teaches that the CVSS score base metrics associated with each of the identified vulnerabilities are used to calculate risk tuples that include “a composite CVSS score component and an aggregate CVSS score component for each program and each host.” EX1005 (Elder), ¶48. The composite score “takes the highest accessibility score and the highest impact score across all the vulnerabilities associated with a software program for a machine and calculates the composite CVSS score from that.” *Id.* Elder teaches that the composite score provides “a measure of severity that reflects that only a single exploited vulnerability is needed by an attacker to achieve the effects of compromise and vulnerabilities associated with a product are not independent.” *Id.* The aggregate score “provides a summation of the CVSS scores of all vulnerabilities associated with a program or a host” and thus “provides a measure of the number of options that are available to an attacker for a program or a host.” *Id.*, ¶49. A POSA would have understood that each of the composite score and aggregate score identifies one or more of the known

and identified vulnerabilities associated with a given software application or a given workload susceptible to attack from outside the workload.

120. As further evidence that the CVSS score base metrics enable Elder's system to identify one or more of the known and identified vulnerabilities susceptible to attack from outside the workload, Elder also teaches calculating a risk quantification score "based on attack graph generation." *Id.*, ¶50. "For example, for a given piece of software, the attack graph can be constructed with the nodes being software states and the edges being the vulnerabilities present in the software." *Id.* The graph can then be used to determine "the least accessibility needed (e.g., accessibility of one of the start nodes) and the highest impact (e.g., impact of one of the final nodes)." *Id.* The graph can also be "arranged as a linked list where the start node has the least accessib[ility] needed of all vulnerabilities and the final node has the highest impact of all vulnerabilities." *Id.*, ¶51.

121. Finally, while Elder does not expressly disclose additionally using at least one port as part of identifying the known and identified vulnerabilities susceptible to attack from outside the workload, as I explained above, a POSA would have used such information in view of Kim. *Supra*, §XII.A. For example, Kim teaches that, to assess the vulnerabilities present on a host, "collection target information" is obtained, which can include "network port status information" of the

host. EX1006 (Kim), ¶¶15, 50. A POSA would have understood that network port status information referred to the status of a network port on the host (e.g., whether it was open or closed). As I explained above, the status of ports was known to inform whether a vulnerability present on an installed software application was susceptible to attack. *See supra*, §§VIII.B.2, XII.A. For example, if a port was open through which a vulnerable application would receive network data, then a POSA would have understood that the vulnerable application would be more susceptible to an attack than if the port were closed. *See, e.g.*, EX1051 (Nyanchama), ¶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 (Cole), ¶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 (Murchison), 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”). Thus, in view of Kim, a POSA would have used at least one port to provide additional information as to the software vulnerabilities on an asset that are susceptible to outside attack. 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.

122. It is therefore my opinion that the combination of Elder and Kim teaches all aspects of elements 1.6, 12.6, and 18.6.

123. Accordingly, it is my opinion that independent claims 1, 12, and 18 were obvious.

C. Dependent Claims

1. Claims 2-3 and 13-14

124. Claims 2 and 13 depend from 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.” As I explain below, Elder alone, or in combination with Kim, teaches these limitations.

125. Elder teaches, in response to the identified vulnerabilities, reporting the calculated risk scores, along with an identification of the software product, host, and vulnerability contributing the most risk to an enterprise, to an administrator. EX1005, ¶¶32 (system includes a report server), 63-68, 79, Fig. 7, claim 1. For example, when the system computes a risk score (e.g., a composite or aggregate risk

score) for a software product or host, the risk score is automatically transmitted to the administrator. *Id.*, ¶¶63-64 (“[A] composite risk score for the host and each software product of the host is determined and reported based on the plurality of vulnerability scores.”), Fig. 7; *see also id.*, ¶¶6 (“[E]mbodiments of the present invention provide a system and method for automatically calculating risk based on client configuration data and vulnerability information.”), 8, 46 (“Risk analysis module 608 automatically provides detailed risk analysis, quantification, and prioritization based on each configuration element (e.g., programs and the operating system) installed on each host”), 85. Elder explains this mechanism provides an indication to the administrator as to where remediation efforts (e.g., patches or upgrades) should be focused. EX1005 (Elder), ¶¶49, 54 (“Embodiments of the present invention thus perform automatic risk analysis ... that allows a user to be presented with actionable intelligence that can be used to prioritize remediation activities”), 69 (“When a major new vulnerability is announced, embodiments of the present invention allow an administrator to recalculate the risk scores to understand how critical and widespread this vulnerability is in his or her enterprise and the level of importan[ce] of remediation activities related to the new vulnerability. This thereby indicates which vulnerability should be patched first.”). Kim similarly teaches “notifying virtual machine users of the inspection results” to allow a system

operator to use the results for security management purposes. EX1006 (Kim), ¶¶54-57. Thus, Elder alone, or in combination with Kim, teaches implementing a remedial action in response to the identified one or more vulnerabilities in the form of transmitting an alert (e.g., automatically notifying calculated risk scores and associated risk information) to a device associated with an administrator.

126. It is therefore my opinion that claims 2-3 and 13-14 were obvious.

2. Claims 4 and 15

127. 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 this limitation. In particular, as detailed below, Kim teaches querying a cloud provider API to determine “network accessibility information related to the workload.” In addition, 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 and identified vulnerabilities that are susceptible to attack from outside the workload.

128. First, in addition to obtaining network port status information of a host, Kim further teaches querying the cloud provider API to obtain other network-related information, including information regarding the network accessibility of the host (e.g., virtual machine). For example, Kim teaches that the cloud provider API can be queried for “network connection information” of the host (i.e., workload) as part of the “collection target information.” EX1006 (Kim), ¶22. A POSA would have understood that such information would include the status of the workload’s network connection. *See id.*, ¶21 (vulnerability inspection may include “network connection status”). Kim teaches that the network connection information is determined by examining data from the virtual disk or virtual memory of the virtual machine (i.e., data sources associated with the workload). *Id.*, ¶¶14-15 (collection target information may be obtained through a virtual disk read/write API or virtual memory read API), 22 (collection target information may include “network connection information”).

129. In addition, as I explained above, a POSA would have had reason to use additional network-related information as taught by Kim as part of the software

risk analysis assessment taught by Elder in order to identify the known and identified vulnerabilities that are susceptible to attack from outside the workload. *Supra*, §§XII.A, XII.B.8. A POSA would have understood that, in addition to network port status information, information regarding the network connection of the workload (e.g., whether the asset was connected to the Internet) would also beneficially provide additional context to assessing whether a vulnerability present on the workload was susceptible to outside attack. *See, e.g.*, EX1045 (Mohanty), 3:63-67, 4:38-46, 5:47-48 (assessing whether a 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, would generate a higher contextual score 216 than would be the case for an asset that does not have Web-connectivity.”).

130. It is therefore my opinion that claims 4 and 15 were obvious.

3. Claims 5 and 16

131. 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 this limitation.

132. As I noted above, Elder explains that its system is configured to identify software applications installed on the host by obtaining host configuration information from files stored on the host. *Supra*, §XII.B.4. Elder specifically teaches that the host configuration information may be obtained from “dynamic link library (DLL) files” (i.e., libraries). EX1005 (Elder), ¶34. As I also explained above, Elder describes enumerating the installed software applications and their associated versions as CPEs in order to query the vulnerability database. *Id.*, ¶¶37-38; *supra*, §XII.B.5. A POSA would have understood that, to generate the CPEs, the system’s processor would be configured to extract data from the library files and then analyze the extracted data to identify software applications installed on the host, which can then be formatted into a standard CPE enumeration. See EX1018 (Price), ¶32 (“[A] machine image reader 282 can be provided for parsing data in retrieved virtual machine images to determine attributes of a virtual machine that can be scanned, checked, and verified in accordance security tasks performed by the security tool 205.”); *supra*, §XII.B.2.

133. It is therefore my opinion that claims 5 and 16 were obvious.

4. Claims 6 and 17

134. 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.” As I explain below, Elder teaches this limitation.

135. I note claim 17 recites that “the at least one processor” is configured to identify a version of the installed software application, but claim 17 is a method claim that depends from claim 12 (via claim 16), which does not recite a processor. Nevertheless, I explain below how Elder teaches a method in which a version of the installed software application is identified for vulnerability analysis.

136. I previously explained how Elder’s system is configured to identify a version of the installed software application in order to match the installed software application to known vulnerabilities stored in a vulnerability database. *Supra*, §XII.B.5; EX1005 (Elder), ¶¶34, 37-39. I also explained how Elder’s system includes a processor configured to execute the steps recited in claim 1 and its dependent claims. *Supra*, §XII.B.2. The same processor would also perform these steps as part of the method claimed by claim 17.

137. It is therefore my opinion that claims 6 and 17 were obvious.

5. Claim 8

138. Claim 8 depends from claim 1 and further recites “wherein the data structure includes aggregated vulnerability data.” Elder teaches this limitation.

139. As I explained above, Elder teaches that the system “maintains an updated database of vulnerabilities (e.g., such as that provided by the National

Vulnerability Database (NVD), fix databases, etc.).” EX1005 (Elder), ¶29; *supra*, §XII.B.6. “The database catalogs vulnerabilities associated with configuration elements including operating systems, applications, and particular versions of each that are vulnerable.” EX1005 (Elder), ¶29. Thus, Elder’s vulnerability database includes aggregated vulnerability data, as recited.

140. It is therefore my opinion that claim 8 was obvious.

6. Claim 9

141. 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 this limitation.

142. Elder teaches the vulnerability database may comprise vulnerability data “provided by the National Vulnerability Database (NVD),” which is a third-party vendor. EX1005 (Elder), ¶29; EX1031 (NIST NVD Overview), 1 (NVD is a product of NIST Information Technology Laboratory’s Computer Security Division).

143. It is therefore my opinion that claim 9 was obvious.

7. Claim 11

144. 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.” Elder teaches this limitation.

145. As I explained above, Elder’s vulnerability database maintains aggregated vulnerability data, such as data provided by the NVD. *Supra*, §§XII.B.6, XII.C.5-6. The vulnerabilities listed by the NVD includes those that may be exploited by a malicious attacker (i.e., an exploit). See EX1005 (Elder), ¶48 (discussing risk analysis measuring severity of exploited vulnerability), claim 1. Moreover, the NVD was also known to provide advisories and other security announcements. EX1031 (NIST NVD Overview), 3 (NVD provides resources such as vulnerability advisories, checklists, alerts, vulnerability summaries, and various announcements lists).

146. It is therefore my opinion that claim 11 was obvious.

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

147. In this ground, I discuss a combination of Elder, Kim, and Hufsmith. This builds upon the combination of Elder and Kim that I discussed in Ground 1, further applying Hufsmith’s disclosure of using contextual properties relating to the workload’s execution environment in the network as part of identifying vulnerabilities that are susceptible to outside attack and using aggregated vulnerability data collected by a scanner as part of the risk analysis. In §§XIII.B-E, I explain how these teachings apply to certain limitations of the independent claims and dependent claims 4, 7, 10, and 15. The combination with Hufsmith does not

change my previous explanation in Ground 1 of why the other limitations of 1-6, 8-9, and 11-18 claims were obvious (*supra*, §XII), since the applied teachings of Hufsmith are consistent with the general approach to the security assessment and risk analysis taught by Elder and Kim.

A. Reasons to Combine Elder, Kim, and Hufsmith

148. In my opinion, a POSA would have had reason to combine the teachings of Elder, Kim, and Hufsmith to arrive at claims 1-18.

149. As an initial matter, Elder, Kim, and Hufsmith each describe systems and methods that are analogous to one another and to the '855 patent. EX1007 (Hufsmith), Abstract, ¶¶3-5 (detecting and managing potential vulnerabilities in distributed applications), 8 (CVE analysis), 33 (same), 149 (systems and methods may assess risk for cloud-based resources, such as virtual machines and containers); *supra*, §§XI.A-C (summary of references), XII.A (reasons to combine Elder and Kim). Hufsmith is therefore in the same field as Elder, Kim, and the '855 patent, and its teachings are pertinent to issues addressed by the '855 patent since those teachings involve detecting security problems in computer assets. *See, e.g.*, EX1001, 1:15-18 (“Disclosed embodiments include new systems and methods for securing cloud infrastructure”), 22:3-33 (“FIG. 5 of the disclosed embodiments describes a method of operating a cybersecurity system performing a side scanning function to protect against potential vulnerabilities.”).

150. Furthermore, like the elements of Elder and Kim that I discussed above (*supra*, §§XII.A-B), assessing risk based on environment-specific network accessibility properties relating to a workload and vulnerability data collected by a scanner, as taught by Hufsmith, was well-known and predictable. *Supra*, §VII.B.2; EX1045 (Mohanty), 3:43-67, 5:6-6:1 (“Since the method and system use environment specific details with a threat feed (i.e., a supplier of threat information 208), the derived prioritization is more accurate and meaningful than just the CVSS score 222 and the CVE ID 220.”); EX1048 (Basavapatna), ¶¶96-97 (asset criticality based in part on asset’s IP address and/or geographic location); EX1039 (Giakouminakis), 6:9-7:10 (prioritizing vulnerabilities based on factors such as connection to other assets in the network); EX1046 (Hovor), ¶¶48 (“The system may determine priority information for an asset, type of data store on an asset, or both, using the location of the asset in the network topology.”), 75-77 (asset score based in part on asset scores of other connected assets); EX1059 (Hibbert), ¶¶76-81 (considering network access rights of user of vulnerable application in risk assessment); EX1053 (Veselov), 4:14-18 (scanning system performs “historical analysis of multiple sequential snapshots stored in a database, to identify changes in the virtual computing environment that may have caused or resolved certain vulnerabilities”). Thus, the combination of Elder, Kim, and Hufsmith would have

merely involved using the familiar techniques of determining the environment-specific properties of a workload and scanner-collected vulnerability data in order to assess an identified software vulnerability's susceptibility of attack by an outside malicious attacker.

151. In addition, a POSA would have been motivated to apply Hufsmith's teachings to the combination from Ground 1 to further enhance the identification of vulnerabilities susceptible to outside attack.

152. First, like Elder, Hufsmith teaches scanning cloud-based assets for software vulnerabilities (e.g., by identifying applicable CVEs) and assessing the attack risk of those vulnerabilities by determining the vulnerabilities' CVSS scores. EX1007 (Hufsmith), ¶¶8 (obtaining "a plurality of scanner properties pertaining to a container" that comprise "one or more Common Vulnerabilities and Exposures (CVE)"), 61, 67 (container images or distributed applications are scanned for vulnerabilities), 82, 84 (scanners may scan resources for vulnerabilities "identified in public repositories, such as repositories of CVE or CWE vulnerabilities"), 105. As with Elder, Hufsmith teaches using the base 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. *Id.*, ¶¶113-20 (CVE evaluator evaluates CVE properties of assets to determine one or more

CVE weights, which can be based on a CVSS vector, 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”).

153. Hufsmith goes on to explain that the risk metrics associated with the CVSS scores can be further refined by additionally considering context-specific network and host properties of the asset, such as its execution environment. *Id.*, 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 (tracking context-specific properties, including information about execution environments, of workloads). For example, Hufsmith teaches that the context-specific properties can include network accessibility information related to the asset, such as the asset’s relative location and exposure within the network (e.g., whether the asset is behind a firewall, within an internal network, 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”).

154. A POSA would have appreciated that such environment-specific network accessibility information would improve upon the accuracy of assessing the likelihood that an outside attacker could successfully exploit a software vulnerability present on the host. Indeed, Hufsmith provides several examples that illustrate this point. “For instance, 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,” which would indicate a decreased risk that a vulnerability could be exploited due to the subroutine’s dormancy. EX1007 (Hufsmith), ¶140. As another example, Hufsmith explains that the susceptibility of “a vulnerability to a DDoS attack” to outside attack can be reduced if “the execution environment of a container i[s] behind a company firewall and for internal use,” which limits network access to the asset. *Id.*, ¶141. In this manner, the vulnerabilities that are the most susceptible to outside attack may be more accurately identified. A POSA therefore would have recognized the benefits of improving upon Elder’s risk assessment by incorporating additional network accessibility information, such as environment-specific properties of the workload in the network,

as part of identifying the known and identified vulnerabilities susceptible to attack from outside the workload.

155. Second, Hufsmith also teaches that the risk assessment may take into account historical risk information. EX1007 (Hufsmith), ¶92 (results engine “may adjust weights associated with one or more vulnerabilities based on ... historical information” of the asset). For example, Hufsmith teaches that the contextual properties for a given asset may also include a “score history.” *Id.*, ¶135. The score history “may include one or more historical scores” determined for the asset based on the results of previous risk assessments. *Id.*, ¶¶106, 139. Hufsmith explains that this type of historical data “may 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.*, ¶139. The concept of tracking the results of previous scans of an asset is similarly recognized in Elder. See EX1005 (Elder), ¶¶49 (teaching use of 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 (contemplating recalculation of risk scores “[w]hen a major new vulnerability is announced,” so that an administrator can “understand how critical and widespread this vulnerability is in his or her enterprise and the level of

important[ce] of remediation activities related to the new vulnerability”). Thus, a POSA would have recognized the benefit of tracking historical scan data for an asset, as taught by Hufsmith, as part of a risk assessment such as Elder’s as doing so would allow for greater insight into the overall risk trend for an asset and the effectiveness of remediation efforts.

156. A POSA would have had a reasonable expectation of success in combining the above teachings of Elder, Kim, and Hufsmith to arrive at the claimed invention. As I explained above, the use of environment-specific network accessibility information of a workload as part of a risk analysis was well-understood and routinely used in security systems like Elder and Kim. *See, e.g.*, EX1045 (Mohanty), 3:43-67, 5:6-6:1; EX1048 (Basavapatna), ¶¶96-97; EX1039 (Giakouminakis), 6:9-7:10; EX1046 (Hovor), ¶¶48, 75-77; EX1059 (Hibbert), ¶¶76-81. Similarly, the use of historical scan data to analyze trends in an asset’s risk was also a well-understood and routinely practiced technique, as confirmed by Elder. EX1005 (Elder), ¶¶49, 69; *see also* EX1053 (Veselov), 4:14-18. Therefore, in my opinion, combining the references’ teachings as discussed above would not have presented meaningful technical challenges or produced unexpected results.

B. Claims 1, 12, and 18

157. 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.” As I explain below, the combination of Elder, Kim, and Hufsmith teach these elements.

158. I explained above in Ground 1 how the combination of Elder and Kim teaches using 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 data from an external data source (i.e., CVSS scores and related metrics). *Supra*, §§XII.A, XII.B.8.

159. Similar to Elder and Kim, Hufsmith also teaches scanning a cloud-based asset to identify vulnerabilities introduced by installed applications and using CVSS score metrics to assess the risk that the vulnerability presents to the asset. *See* EX1007 (Hufsmith), ¶¶8, 61, 67, 82, 84, 105, 113-120, 149. As I explained above, Hufsmith teaches that this risk assessment can be further refined by additionally considering environment-specific network accessibility information related to the workload, such as whether the workload is externally exposed, located behind a firewall, or located within an internal network. *Id.*, ¶¶92, 106, 135, 140-41; *supra*, §XIII.A. This type of information would have been understood to be “network

accessibility information” related to a workload that includes “data from an external data source” (e.g., data regarding the asset’s operating environment) and “cloud provider information” (e.g., information regarding network access to the workload), as recited by the claim elements. *Compare* 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.”).

160. As I discussed in §XIII.A, it would have been obvious to apply these teachings with the combination of Ground 1 such that the risk analysis would take into account environment-specific network and host properties relating to the workload in identifying the known and identified software vulnerabilities that are susceptible to attack from outside the workload.

161. It is therefore my opinion that claims 1, 12, and 18 were obvious for these additional reasons.

C. Claims 4 and 15

162. 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.” As I explain below, the combination of Elder, Kim, and Hufsmith teaches this limitation.

163. I explained above in Ground 1 how Kim teaches querying a cloud provider API to determine network accessibility information related to the workload (e.g., network connection information) by examining data sources associated with the workload (e.g., virtual disk and memory). *Supra*, §XII.C.2.

164. Hufsmith also teaches determining network accessibility information related to the workload (e.g., environment-specific network accessibility information) by examining data sources related to the workload. For example, Hufsmith teaches that environment data may be obtained through configuration files associated with the workload. See EX1007 (Hufsmith), ¶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 such data sources because this type of information would have been understood to be accessible through APIs provided by the cloud environment. *See, e.g.*, EX1006 (Kim), ¶22 (information obtained through APIs can include configuration files of virtual machines).

165. It is therefore my opinion that claims 4 and 15 was obvious.

D. Claim 7

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

167. Hufsmith teaches that applications installed on workloads found in a cloud environment may include one or more scripts. For example, Hufsmith explains that, as part of the risk analysis, the composition files of applications installed on the workloads may be scanned. EX1007 (Hufsmith), ¶¶58, 61, 73, 86, 108. Hufsmith explains that these composition files may include scripts. For instance, Hufsmith teaches that scanners used to perform the risk assessment may be selected “for various source code of interpreted languages (e.g., Python™, JavaScript™, and the like).” *Id.*, ¶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.”). In view of Hufsmith’s teachings, a POSA would have understood that many applications would have included scripts because scripts were commonly used programming instructions for executing certain functions of an application. See EX1007 (Hufsmith), ¶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)....”).

168. It is therefore my opinion that claim 7 was obvious.

E. Claim 10

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

170. I explained above in Ground 1 how Elder teaches that its risk analysis system maintains a data structure that includes aggregated vulnerability data (e.g.,

vulnerability database of known software vulnerabilities). *Supra*, §§XII.C.5-7. Hufsmith further teaches maintaining a score database for a scanned workload (e.g., a container). EX1007 (Hufsmith), ¶¶100-02, 105, 205, Fig. 1B. The score database stores “CVE properties 111,” which “may correspond to a set of identified CVE properties determined by a static analysis scanner or a dynamic scanner.” *Id.*, ¶105. Moreover, the score database also includes a “score history 128,” which “may include one or more historical scores” that “may 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. In other words, Hufsmith teaches maintaining, in a database, the CVE properties of a workload identified by a scanner and the previous risk scores calculated based on the vulnerabilities identified from scans of the workload. Thus, Hufsmith teaches a data structure comprising “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 determined by a scanner and the historical risk scores that were derived from the vulnerabilities identified by previous scans).

171. As I discussed in §XIII.A, it would have been obvious to apply these teachings with the combination of Ground 1 such that the risk analysis would take

into account historical scan data relating to the workload to analyze trends in calculated risk scores and effects of remediation efforts. Further, a POSA would have found it obvious that a historical scan database, such as taught by Hufsmith, could readily be included in the same data structure as the vulnerability database taught by Elder. See EX1005 (Elder), ¶¶29, 40 (teaching fix database and vulnerability database may be “part of the same database or data source”). Thus, the combination of Elder, Kim, and Hufsmith teaches a data structure having aggregated vulnerability data that includes data collected by a scanner, as recited.

172. It is therefore my opinion that claim 10 was obvious.

XIV. CONCLUDING STATEMENTS

173. In signing this declaration, I understand that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I acknowledge that I may be subject to cross-examination in this case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

174. I declare that all statements made herein of my knowledge are true, and that all statements made on information and belief are believed to be true, and that these statements were made with the knowledge that willful false statements and the

like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: 11/01/2024

By: /Angelos Stavrou/
Angelos Stavrou, Ph.D.

XV. APPENDIX A – MATERIALS CITED

Exhibit No.	Description
1001	U.S. Patent No. 11,637,855 to Shua
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 Nyanchama

Exhibit No.	Description
1052	U.S. Publication No. 2004/0015728 A1, published January 22, 2004, to Cole
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