

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CROWDSTRIKE, INC.,

Petitioner

v.

SKYSONG INNOVATIONS, LLC,

Patent Owner

Inter Partes Review Case No. IPR2025-01397

U.S. Patent No. 10,313,385

DECLARATION OF EREZ ZADOK, Ph.D.

TABLE OF CONTENTS

<i>I. Educational Background and Qualifications</i>	12
<i>II. Methodology and Materials Considered</i>	24
<i>III. Legal Framework</i>	28
<i>IV. Opinion</i>	35
A. The '385 Patent	35
B. Summary of the '385 Patent's Prosecution History	37
C. Claim Construction	42
D. Level of a Person of Ordinary Skill in the Art	42
<i>V. Background of the Technology</i>	44
A. Web Crawling Using Algorithms	45
B. Retrieving Exploit Data from Dark Net Markets	47
C. Identifying Vulnerabilities Relevant to the Mined Exploits	48
D. Applying Exploits to Determine Attack Patterns	49
E. Identifying a Limited Group of Vulnerabilities to Test	51
F. Testing Exploits in a Protected Environment	53
G. Remediating the Computer System Based on the Results of Testing ...54	
<i>VI. Opinion – Ground 1</i>	56
A. Obviousness over Vallone	56
1. Overview of Vallone	56
2. Claim 1(Pre): “A system for protecting a computer system from attack, the system comprising:”	59
3. Claim 1(a): “a network connection;”	60
4. Claim 1(b): “a processing device; and”	60
5. Claim 1(c): “a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:”	61
6. Claim 1(d): “accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:”	62

7. Claim 1(e): “obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;”	64
8. Claim 1(f): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”	68
9. Claim 1(g): “creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system;”	71
10. Claim 1(h): “applying the set of exploits to the constraint set of the computer system;”	74
11. Claim 1(i): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”	77
12. Claim 1(j): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”	78
13. Claim 8(Pre): “A method for improving a computer device, the method comprising:”	79
14. Claim 8(a): “accessing data comprising dark net information associated with a computer system;”	80
15. Claim 8(b): “obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of the computer system;”	80
16. Claim 8(c): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”	81
17. Claim 8(d)(1): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system,”	81
18. Claim 8(d)(2): “wherein application of the set of exploits on the computer system comprises determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system;”	82
19. Claim 8(e): “analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system; and”	82
20. Claim 8(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”	83
21. Claim 14: “The method of claim 8, wherein altering the configuration of the computer system comprises installing a patch to a software component of the computer system that is exposed by the set of exploits to attack the computer system.”	83

22. Claim 15(Pre): “One or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine, the computer process comprising:”	84
23. Claim 15(a): “obtaining a set of exploits from dark net information, the set of exploits configured to bypass a security feature of a computer system;”	84
24. Claim 15(b): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”	84
25. Claim 15(c): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system;”	85
26. Claim 15(d): “applying the set of exploits to the constraint set of vulnerabilities of the computer system;”	85
27. Claim 15(e): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”	85
28. Claim 15(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”	85
VII. OPINION – GROUND 2	85
A. Obvious to Combine Oberheide, Patsos, and Roytman	86
1. Overview of Oberheide	86
2. Overview of Patsos	89
3. Overview of Roytman	91
4. Claim 1(Pre): A system for protecting a computer system from attack, the system comprising:	93
5. Claim 1(a): a network connection;	94
6. Claim 1(b): a processing device; and	94
7. Claim 1(c): a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:	95
8. Claim 1(d): accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:	96
a. Motivation to Combine Oberheide with Patsos (Dark Net Data)	97
9. Claim 1(e): obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;	103

10. Claim 1(f): applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;.....	104
a. Motivation to Combine Oberheide with Patsos (Exploit Function) ..	107
11. Claim 1(g): creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system	110
a. Motivation to Combine Oberheide with Roytman (Minimum Constraint Set)	115
12. Claim 1(h): applying the set of exploits to the constraint set of the computer system;	118
13. Claim 1(i): analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and.....	120
14. Claim 1(j): altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.....	122
15. Claim 8(Pre): A method for improving a computer device, the method comprising:	123
16. Claim 8(a): “accessing data comprising dark net information associated with a computer system;”	124
17. Claim 8(b): obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of the computer system;	124
18. Claim 8(c): applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;.....	125
19. Claim 8(d)(1): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system, ...	125
20. Claim 8(d)(2): wherein application of the set of exploits on the computer system comprises determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system;	125
21. Claim 8(e): analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system; and.....	126
22. Claim 8(f): altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.....	126
23. Claim 14: “The method of claim 8, wherein altering the configuration of the computer system comprises installing a patch to a software	

component of the computer system that is exposed by the set of exploits to attack the computer system.”	126
24. Claim 15(Pre): “One or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine, the computer process comprising:”	127
25. Claim 15(a): “obtaining a set of exploits from dark net information, the set of exploits configured to bypass a security feature of a computer system;”	127
26. Claim 15(b): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”	128
27. Claim 15(c): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system;”	128
28. Claim 15(d): “applying the set of exploits to the constraint set of vulnerabilities of the computer system;”	128
29. Claim 15(e): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”	128
30. Claim 15(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”	128
VIII. Opinion – Ground 3	129
A. Obvious to Combine Oberheide, Patsos, Roytman, and Juels	129
1. Overview of Juels	129
2. Claim 2: The system of claim 1, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.....	130
a. Motivation to Combine Oberheide with Juels (Exploit Costs)	132
3. Claim 3: The system of claim 2, wherein the processing device further performs the operation of: aligning Internet data with the constraint set of vulnerabilities based on information obtained about the set of exploits.....	134
4. Claim 4: The system of claim 3, wherein obtaining a set of exploits through the network connection comprises: applying one or more algorithms to obtain the set of exploits.....	136
5. Claim 5: The system of claim 4, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and	

obtains the set of exploits constrained by the determined cost to obtain the set of exploits.	138
a. Motivation to Combine Oberheide with Juels (Exploit Budgets)	140
6. Claim 6: The system of claim 5, wherein obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.....	141
7. Claim 9: The method of claim 8, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.....	145
8. Claim 10: The method of claim 9, wherein obtaining a set of exploits comprises: applying one or more algorithms to obtain the set of exploits based on the constraint set of vulnerabilities of the computer system from one or more darknet marketplaces on the Internet.	145
9. Claim 11: The method of claim 10, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtains the set of exploits constrained by the determined cost to obtain the set of exploits.	147
10. Claim 12: The method of claim 11, where obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.....	147
11. Claim 16: The one or more non-transitory computer-readable storage media of claim 15, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.....	147
12. Claim 17: The one or more non-transitory computer-readable storage media of claim 15, wherein obtaining a set of exploits comprises applying one or more algorithms to obtain the set of exploits from one or more darknet marketplaces on the Internet.....	147
<i>IX. Opinion – Ground 4</i>	148
A. Obvious to Combine Oberheide, Patsos, Roytman and Wolff	148
1. Overview of Wolff.....	148
2. Claim 7: “The system of claim 1, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.”.....	151
a. Motivation to Combine Oberheide with Wolff.....	152

3. Claim 13: “The method of claim 8, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.”.....154

X. CONCLUSION.....155

Challenged Claims

Claim Designation	Claim Language
Claim 1(Pre)	A system for protecting a computer system from attack, the system comprising:
Claim 1(a)	a network connection;
Claim 1(b)	a processing device; and
Claim 1(c)	a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:
Claim 1(d)	accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:
Claim 1(e)	obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;
Claim 1(f)	applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;
Claim 1(g)	creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system;
Claim 1(h)	applying the set of exploits to the constraint set of the computer system;
Claim 1(i)	analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and
Claim 1(j)	altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.

Claim Designation	Claim Language
Claim 2	The system of claim 1, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.
Claim 3	The system of claim 2, wherein the processing device further performs the operation of: aligning Internet data with the constraint set of vulnerabilities based on information obtained about the set of exploits.
Claim 4	The system of claim 3, wherein obtaining a set of exploits through the network connection comprises: applying one or more algorithms to obtain the set of exploits.
Claim 5	The system of claim 4, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtains the set of exploits constrained by the determined cost to obtain the set of exploits.
Claim 6	The system of claim 5, where obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.
Claim 7	The system of claim 1, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.
Claim 8(Pre)	A method for improving a computer device, the method comprising:
Claim 8(a)	accessing data comprising dark net information associated with a computer system;
Claim 8(b)	obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of the computer system;
Claim 8(c)	applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;
Claim 8(d)(1)	creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system,

Claim Designation	Claim Language
Claim 8(d)(2)	wherein application of the set of exploits on the computer system comprises determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system;
Claim 8(e)	analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system; and
Claim 8(f)	altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.
Claim 9	The method of claim 8, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.
Claim 10	The method of claim 9, wherein obtaining a set of exploits comprises: applying one or more algorithms to obtain the set of exploits based on the constraint set of vulnerabilities of the computer system from one or more darknet marketplaces on the Internet.
Claim 11	The method of claim 10, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtains the set of exploits constrained by the determined cost to obtain the set of exploits.
Claim 12	The method of claim 11, where obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.
Claim 13	The method of claim 8, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.
Claim 14	The method of claim 8, wherein altering the configuration of the computer system comprises installing a patch to a software component of the computer system that is exposed by the set of exploits to attack the computer system.

Claim Designation	Claim Language
Claim 15(Pre)	One or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine, the computer process comprising:
Claim 15(a)	obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of a computer system;
Claim 15(b)	applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;
Claim 15(c)	creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system;
Claim 15(d)	applying the set of exploits to the constraint set of vulnerabilities of the computer system;
Claim 15(e)	analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and
Claim 15(f)	altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.
Claim 16	The one or more non-transitory computer-readable storage media of claim 15, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.
Claim 17	The one or more non-transitory computer-readable storage media of claim 15, wherein obtaining a set of exploits comprises applying one or more algorithms to obtain the set of exploits from one or more darknet marketplaces on the Internet.

I, Erez Zadok, Ph.D., declare as follows:

1. I am over the age of 21 and am competent to make this declaration. I make this declaration based on facts and matters within my own knowledge and on information provided to me by others, and, if called as a witness, I could and would competently testify to the matters set forth herein.

I. EDUCATIONAL BACKGROUND AND QUALIFICATIONS

2. My *curriculum vitae*, which includes a more detailed summary of my background, experience and publications, is attached as EX1045.

3. I am a Professor in the Computer Science Department at Stony Brook University (part of the State University of New York (“SUNY”) system). I direct the File-systems and Storage Lab (FSL) at Stony Brook’s Computer Science Department. My research interests include file systems and storage systems, operating systems, transactional systems including database technologies, information technology and system administration, security/privacy and information assurance, networking, energy efficiency, performance and benchmarking, virtualization, cloud systems, compilers, applied machine learning, and software engineering.

4. I studied at a professional high school in Israel, focusing on electrical engineering (“EE”), and graduated in 1982. I spent one more year at the high school’s college division, receiving a special Certified Technician’s degree in EE. I

then went on to serve in the Israeli Defense Forces for three years (1983–1986). I received my Bachelor of Science degree in computer science (“CS”) in 1991, my Master’s degree in CS in 1994, and my PhD in CS in 2001—all from Columbia University in New York.

5. When I began my undergraduate studies at Columbia University, I also started working as a student assistant in the various campus-wide computer labs, eventually becoming an assistant to the head labs manager, who was managing all public computer labs on campus. During that time, I also became more involved with research within the CS Department at Columbia University, conducting research on operating systems, file and storage systems, distributed and networked systems, security, and other topics. I also assisted the CS department’s computer administrators in managing the department’s computers, which included storage, IT, networking, and cyber-security related duties.

6. In 1991, I joined Columbia University’s CS department as a full-time systems administrator, studying towards my MS degree part-time. My MS thesis topic is related to file system reliability, fault tolerance, replication, and failover in mobile networked storage systems using file virtualization. My main duties as a systems administrator involved installing, configuring, and managing many networked servers, proxies, and desktops running several operating systems, as well as network devices setup; this included many software and hardware upgrades,

device upgrades, and BIOS firmware/chipset updates/upgrades. My duties also included ensuring reliable, secure, authenticated access to networked systems/storage and licensed software, as well as software updates, security and bug fixes. Examples of servers and their protocols included email (SMTP), file transfer (FTP), domain names (DNS), network file systems (NFS), network news systems (NNTP), and Web (HTTP).

7. In 1994, I left my systems administrator position to pursue my doctoral studies at Columbia University. My PhD thesis topic was on versatile file system development using stackable (virtualized) file systems, with examples in the fields of security and encryption, efficiency, reliability, and failover. I continued to work part-time as a systems administrator at the CS department, and eventually I was asked to serve as manager to the entire information technology (“IT”) staff. From 1991 to 2001, I was also a member of the faculty-level Facilities Committee that oversaw all IT operations at the CS department.

8. As part of my PhD studies at Columbia, I collaborated on projects to develop advanced AI-like techniques to detect previously unknown viruses (a.k.a. “zero-day malware”), using data mining and rule-based detection. This work led to several highly cited papers (over 1,700 citations for one of the papers alone) and two patents. I also became a Teaching Assistant (“TA”) for a first-ever Computer

Security course given at Columbia University's CS department with Dr. Matt Blaze as instructor.

9. From 1990 to 1998, I consulted for SOS Corporation and HydraWEB Technologies as a systems administrator and programmer, managing data storage use and backup/restore duties, databases, web servers, as well as information assurance and cyber-security (*e.g.*, malware protection, software licensing). From 1994 to 2000, I led projects at HydraWEB Technologies and then became the Director of Software Development—overseeing the development of several products and appliances such as stateful firewalls and HTTP load-balancers, utilizing network-virtualization and high-availability techniques. From 2009 to 2019, I have consulted for Packet General Networks, a startup specializing in secure, virtualized, network storage and applications' data security in the cloud.

10. In 2001, I joined the faculty of Stony Brook University, a position I have held since that time. In 2002, I joined the Operations Committee, which oversees the IT operations of the CS department at Stony Brook University. From 2006 to 2010, I was the Director of IT Operations of the CS department. My day-to-day duties included setting policies regarding computing, hiring and training new staff, assisting any staff with topics of my specialty, defining requirements for new software/hardware, and purchasing. From 2010 to 2015, I had served as the Co-Chair to the Operations Committee. From 2016 to 2019, I oversaw the IT Operations

as the Chair of the Operations Committee. A significant component of these duties included defining and helping implement policies for data management, to ensure the security of users and their data, and data reliability and availability, while minimizing the inconvenience and performance impact to users. I personally helped setup and maintain an initial virtual-host infrastructure in the department. Since late 2019, I've been a member of the department's Executive Committee that also oversees all IT operations.

11. In 2017, I became the department's Graduate Academic Adviser, advising all Master students (over 400 annually on average) and many other graduate students on an assortment of academic matters. In 2024, I took over as the department's Graduate Program Director, overseeing the entire graduate CS program (700-800 students annually on average).

12. Since 2001, I have personally configured and managed my own research lab's network. This includes setting up and configuring multiple storage systems (*e.g.*, NFS, CIFS/SMB, NAS), virtual and physical environments, applications such as database (*e.g.*, MySQL, PostgreSQL), Web servers (*e.g.*, Apache), and mail servers; user access control (*e.g.*, NIS, LDAP), backups and restores, snapshot policies, and more. I have personally installed, configured, changed, replaced parts, and upgraded components in numerous devices including mobile devices, laptops, desktops, and servers, both physical and virtual.

13. Since 1995, I have taught courses on operating systems, storage and file systems, advanced systems programming in Unix/C, systems administration, data structures, data/software security, and more. My courses often use storage, file systems, distributed systems, and system/network security as key teaching principles and practical examples for assignments and projects. I have taught these concepts and techniques to my students, both to my direct advisees as well as in my courses. For example, in my graduate Operating Systems course, I often cover Linux's kernel mechanisms to protect users, applications, and data files, virtual file systems, as well as distributed storage systems (*e.g.*, NFS). And in the System Administration undergraduate course, I covered many topics such as networking, storage, backups, and configuring complex applications such as mail, web, and database servers.

14. My research often investigates computer systems from many angles: security, efficiency, energy use, scalability, reliability, portability, survivability, usability, ease-of-use, versatility, flexibility, and more. My research gives special attention to balancing five often-conflicting aspects of computer systems: performance, reliability, energy use, security, and ease-of-use.

15. Since joining Stony Brook University in 2001, my group in the File-systems and Storage Lab ("FSL") has developed many file systems and operating system extensions; examples include a highly-secure cryptographic file system, a portable copy-on-write ("COW") versioning file system, a tracing file system useful

to detect intrusions, a replaying file system useful for forensics, a snapshotting and sandboxing file system, a namespace unification file system (that uses stackable, virtualized, file-based COW), an anti-virus file system, an integrity-checking file system, a load balancing and replication/mirroring file system, network file system extensions for security and performance, distributed secure cloud-based storage systems, transactional key-value stores and file systems, OS-level embedded databases, a compiler to convert user-level C code to in-kernel efficient yet safe code, GCC plugins, stackable file system templates, and a Web-based backup system. Many of these projects used one form of virtualization or another (storage, network, host, etc.). I continue to maintain and release newer versions of some of these file systems and software.

16. I have published over 120 refereed publications (in ACM, IEEE, USENIX, and more). To date, my publications have been cited more than 10,000 times (as per Google Scholar as of August 11, 2025). My papers cover a wide range of related technologies such as file systems, storage systems, transactional systems, security, clouds and virtualization, performance benchmarking and optimization, energy efficiency, system administration, web systems, and more. I also published a book titled “Linux NFS and Automounter Administration” (Sybex, 2001), covering systems administration topics related to network storage and data security.

17. Some of my research has led to public software releases that have been used worldwide. I have publicly maintained the Amd Berkeley Automounter in a package called “am-utils” since 1992; this software helps administrators manage the multitude of file system mounts on dozens of different Unix systems, especially helping to automate access to multiple NFS/NAS storage volumes. Since 1997, I have maintained and released several stackable (virtualized) file system software projects for Linux, FreeBSD, and/or Sun Solaris, in a package called FiST. One of my stackable file system encryption projects, called Cryptfs, became the basis for IBM’s public release of eCryptfs, now part of Linux. Packet General Networks, for whom I have provided consulting services between 2009 and 2019, licensed another encryption file system called Ncryptfs. Another popular file system released in 2003, called Unionfs, offers virtual namespace unification, transparent shadow copying (a.k.a. copy-on-write or COW), file system snapshotting (*e.g.*, useful for forensics and disaster recovery), and the ability to save disk space by sharing a read-only copy of data across several computers, among other features.

18. My research and teaching make extensive use of data security features. For example, each time I taught the graduate operating system course, the first homework assignment includes the creation of a new system call that performs new or added functionality, often for encrypting a file or verifying its integrity; many of my other assignments cover topics of user/process access control, anti-virus filtering,

and more. Since 2001, over 1,000 graduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

19. Moreover, in an undergraduate course titled “Advanced Systems Programming in Unix/C,” I cover many topics of system security and vulnerabilities, such as the structure of UNIX processes, and memory segments such as the heap and stack. This course covers details of several hundred Linux system calls. Often, the first assignment for this course is to develop a tool to encrypt/decrypt files using advanced ciphers, use digital signatures to certify the cipher keys used, and reliably recover files in case of failures. Since 2001, several hundred undergraduate students were exposed to these principles directly through my teaching and research at Stony Brook University.

20. In another undergraduate course, System Administration, I taught network configuration, security, and storage configuration and reliability. In a special topics course on Storage Systems, I covered many topics such as data deduplication, RAID, transactional storage, storage hardware including modern Flash based ones, virtual storage, backup/restore, snapshots and continuous data protection (“CDP”), NAS and SAN, and NFS.

21. Overall, in addition to the aforementioned experience, my technical experience relevant to the ’385 Patent at the time of the alleged invention included the following: I studied, researched, developed code, published, taught, released

software, and/or worked on cases covering the following techniques and technologies: encryption and integrity; intrusion detection and investigation using network packet sniffers (*e.g.*, tcpdump), system call tracers (*e.g.*, strace), and library call tracers; post-intrusion inspection, cleanup, and analysis (*e.g.*, Unix workstations and/or Windows PCs); inspecting systems suspected of active intrusions for hidden files, unknown processes, and unexpected open network sockets; using network tools such as port scanner to look for vulnerabilities; write scripts using various tools to scan every host in a network for vulnerabilities; using and monitoring file checksum tools (*e.g.*, tripwire) for signs of Trojan files and other break-ins; installing, updating, and monitoring anti-malware tools; updating system software regularly and automatically to close security vulnerabilities; change system configurations (*e.g.*, application config files, system password files) to close security vulnerabilities; monitor discussion forums, both public and private, for vulnerability reports, test such reports in confined/safe settings, and assess the risk level of each vulnerability; configure firewalls to control incoming and outgoing traffic; analyzing source code for bugs and vulnerabilities, fixing code, deploying new software versions, and reporting bugs to vendors and/or publicly; multi-modal cost modeling (*e.g.*, throughput, latency, CPU usage, memory usage, energy consumption, security level, and economic cost models and functions); multi-dimensional optimizations using Pareto Frontier optimizations; developed an anti-virus layered file system; and

developed and patented AI-based techniques to detect zero-day malware, which included collecting and analyzing malware from “dark net” and public sources.

22. My research has been supported by many federal and state grants as well as industry awards, including an NSF CAREER award, two IBM Faculty awards, two NetApp Faculty awards, a Western Digital award, a Facebook award, several Dell-EMC awards, and several equipment gifts. I received the 2008 SUNY Chancellor’s Excellence in Teaching award, and the 2022 SUNY Chancellor’s Award for Excellence in Scholarship and Creative Activities (both awards can be given only once in a lifetime). In 2021, I was named an ACM Distinguished Member for “Outstanding Scientific Contributions to Computing.”

23. My service record to the community includes serving as the co-chair for the USENIX Annual Technical Conference in 2020 (ATC’20); serving as the co-chair for USENIX File and Storage Technologies (FAST’15) in 2015 and on the FAST Conference Steering Committee from 2015 to 2023; serving on the ACM HotStorage Steering Committee since 2021; and serving as the co-chair in 2012 and on the Steering Committee of the ACM SYSTOR conference since 2012. I have served as an Associate Editor to the ACM Transactions on Storage (“TOS”) journal from 2009 to 2022; in 2022, I was named the Editor-in-Chief for ACM’s TOS journal.

24. I am a named inventor on four patents, two titled “Systems and Methods

for Detection of New Malicious Executables” (U.S. Patent No. 7,487,544, issued February 3, 2009; and U.S. Patent No. 7,979,907, issued July 12, 2011); and two more titled “Multi-Tier Caching,” (U.S. Patent No. 9,355,109, issued May 31, 2016; and U.S. Patent 9,959,279, issued May 1, 2018).

25. I have been disclosed as a testifying expert in 24 cases (including *inter partes* review (“IPR”) proceedings) in the past four years. I have been deposed 14 times and testified in trial twice.

26. I have personal knowledge of the facts and opinions discussed in this declaration and believe them to be true. If called upon to do so, I would testify competently thereto. I have been warned that willful false statements and the like are punishable by fine or imprisonment, or both. I have had no contact with any of the named inventors of the ’385 Patent.

27. My consulting company, Zadoks Consulting Services, is being compensated for my time at my current standard consulting rate. I am also being reimbursed for expenses that I may incur during the course of this work. My compensation is not contingent upon the results of my study and analysis, the substance of my opinions, or the outcome of any proceeding involving the Challenged Claim(s).

28. I have no financial interest in the Petitioner. I have no financial interest in the outcome of this matter or in any litigation involving the ’385 patent.

II. METHODOLOGY AND MATERIALS CONSIDERED

29. I have relied upon my education, knowledge and experience with the technology of communication networks, as well as the other materials as discussed in this declaration in forming my opinions.

30. For this work, I have been asked to review U.S. Patent No. 10,313,385 (“’385 Patent”) (EX1001) including the specification and claims, and the ’385 Patent’s prosecution history (“’385 File History”) (EX1002). In developing my opinions relating to the ’385 Patent (EX1001), I have considered the materials cited or discussed herein, including those listed below.

Exhibit	Description
1001	U.S. Patent No. 10,313,385 to Shakarian <i>et al.</i> (“’385 Patent”)
1002	File History of U.S. Patent No. 10,313,385 (“’385 Patent File History”)
1004	U.S. Patent Application Publication No. 2016/0134653 to Vallone <i>et al.</i> (“Vallone”)
1005	U.S. Patent Application Publication No. 2014/0245450 to Oberheide <i>et al.</i> (“Oberheide”)
1006	Dimitrios Patsos, Sarandis Mitropoulos, and Christos Douligeris, <i>Expanding topological vulnerability analysis to intrusion detection through the incident response intelligence system</i> , Information Management & Computer Security, Vol. 18, No. 4, 291-309 (2010) (“Patsos”)
1007	U.S. Patent Application Publication No. 2015/0237065 to Roytman <i>et al.</i> (“Roytman”)
1008	U.S. Patent No. 9,471,777 to Juels <i>et al.</i> (“Juels”)
1009	U.S. Patent Publication No. 2013/0198848 to Wolff (“Wolff”)
1010	U.S. Patent Application Publication No. 2014/0137257 to Martinez <i>et al.</i> (“Martinez”)
1011	Karen Scarfone, Murugiah Souppaya, Amanda Cody, and Angela Orebaugh, <i>Technical Guide to Information Security Testing and Assessment</i> , NIST Special Publication 800-115 (Sept. 2008) (“Scarfone”)

Exhibit	Description
1012	Microsoft Computer Dictionary, Microsoft Press, 4th ed. 201 (1999) (“Microsoft”)
1013	Assane Gueye and Vladimir Marbukh, <i>A Game-Theoretic Framework for Network Security Vulnerability Assessment and Mitigation</i> , In International Conference on Decision and Game Theory for Security (GameSec), (2012) (“Gueye”)
1014	Ritesh Kumar Tiwari and Kamalakar Karlapalem, <i>Cost Tradeoffs for Information Security Assurance</i> , Workshop on the Economics of Information Security (2005) (“Tiwari”)
1015	Christopher Olston and Marc Najork, <i>Web Crawling</i> , Foundations in Information Retrieval, Vol. 4, No. 3, 175-246 (2010) (“Olston”)
1016	Md. Abu Kausar, V.S. Dhaka, and Sanjeev Kumar Singh, <i>Web Crawler: A Review</i> , International Journal of Computer Applications, Vol. 63, No. 2, 31-36 (2013) (“Kausar”)
1017	Seyed M. Mirtaheri, Mustafa Emre Dincturk, Salman Hooshmand, Gregor V. Bochmann, Guy-Vincent Jourdan, and Iosif Viorel Onut, <i>A Brief History of Web Crawlers</i> , arXiv, https://doi.org/10.48550/arXiv.1405.0749 (2014) (“Mirtaheri”)
1018	Tianjun Fu, Ahmed Abbasi, and Hsinchun Chen, <i>A Focused Crawler for Dark Web Forums</i> , Journal of the American Society for Information Science and Technology, Vol. 61, No. 6, 1213-1231 (2010) (“Fu”)
1019	Jiang-Ming Yang, Rui Cai, Chunsong Wang, Hua Huang, Lei Zhang, and Wei-Ying Ma, <i>Incorporating Site-Level Knowledge for Incremental Crawling of Web Forums: A List-wise Strategy</i> , Proceedings of the 15 th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2009) (“Yang”)
1020	Jorge Lucángeli Obes, Carlos Sarraute, and Gerardo Richarte, <i>Attack Planning in the Real World</i> , arXiv:1306.4044 [cs.CR] (June 19, 2013) (“Obes”)
1021	Charlie Miller, <i>The legitimate vulnerability market: inside the secretive world of 0-day exploit sales</i> , Workshop on the Economics of Information Security (May 6, 2007) (“Miller”)
1022	Michael Chertoff and Toby Simon, <i>The Impact of the Dark Web on Internet Governance and Cyber Security</i> , Global Commission on Internet Governance Paper Series No. 6 (Feb. 2015) (“Chertoff”)

Exhibit	Description
1023	Marco Carvalho and Choh Man Teng, <i>Automatic Discovery of Attack Messages and Pre- and Post-Conditions for Attack Graph Generation</i> , Proceedings of the International Conference on Information-Warfare and Security, 378-387 (2010) (“Carvalho”)
1024	Sushil Jajodia, Steven Noel, and Brian O’Berry, <i>Topological Analysis of Network Attack Vulnerability</i> , Managing Cyber Threats, 247-266 (2005) (“Jajodia”)
1025	U.S. Patent Application Publication No. 2006/0021029 to Brickwell <i>et al.</i> (“Brickwell”)
1026	Shinsuke Miwa, Toshiyuki Miyachi, Masashi Eto, Masashi Yoshizumi, and Yoichi Shinoda, <i>Design and Implementation of an Isolated Sandbox with Mimetic Internet used to Analyze Malwares</i> , DETER (2007) (“Miwa”)
1027	Vincent Nicomette, Mohamed Kaaniche, Eric Alata, and Matthieu Herrb, <i>Set-up and Deployment of a High-Interaction Honeypot: Experiment and Lessons Learned</i> , Journal of Computer Virology, Vol. 7, No. 2, 143-157 (2011) (“Nicomette”)
1028	U.S. Patent Application Publication No. 2015/0040230 to Oliphant <i>et al.</i> (“Oliphant”)
1029	Joint Task Force, <i>Security and Privacy Controls for Federal Information Systems and Organizations</i> , NIST Special Publication 800-53 (2013) (“Joint Task Force”)
1030	Plaintiff’s Preliminary Infringement Contentions, <i>Skysong Innovations, LLC v. CrowdStrike, Inc., CrowdStrike Holdings, Inc.</i> , C.A. No. 7:25-cv-00040 (2025) (“Infringement Contentions”)
1031	Declaration of June Munford (“Munford Dec.”)
1032	Europol European Law Enforcement Agency, <i>Review</i> , General Report on Europol Activities (2014) (“Europol”)
1033	U.S. Patent Application Publication No. 2016/0259943 to Murthy <i>et al.</i> (“Murthy”)
1034	Yulei Zhang, Shuo Zeng, Chun-Neng Huang, Li Fan, Ximing Yu, Yan Dang, Catherine A. Larson, Dorthy Denning, Nancy Roberts, and Hsinchun Chen, <i>Developing a Dark Web Collection and Infrastructure for Computational and Social Sciences</i> , Proceedings of the 2010 IEEE International Conference on Intelligence and Security Informatics (2010) (“Zhang”)

Exhibit	Description
1035	Michael Felderer and Ina Schieferdecker. <i>A taxonomy of risk-based testing</i> , International Journal on Software Tools for Technology Transfer, Vol. 16, Issue 5, 559-568 (July 23, 2014) (“Felderer”)
1036	Massimiliano Albanese, Sushil Jajodia and Steven Noel, <i>Time-Efficient and Cost-Effective Network Hardening Using Attack Graphs</i> , In Proceedings of the 42nd Annual IEEE/IFIP International Conference on Dependable Systems and Networks (June 2012) (“Albanese”)
1037	U.S. Patent No. 9,082,126 to Nielsen <i>et al.</i> (“Nielsen”)
1038	Luca Allodi and Fabio Massacci, <i>Comparing Vulnerability Severity and Exploits Using Case-Control Studies</i> , ACM Transactions on Information and System Security, Vol. 17, No. 1, Article 1 (Aug. 2014) (“Allodi”)
1039	U.S. Patent Application Publication No. 2011/0191854 to Giakouminakis <i>et al.</i> (“Giakouminakis”)
1040	Gabriele Gianini, Marco Cremonini, Andrea Rainini, G. Lena Cota, and L. Ghemmogne Fossi, <i>A Game Theoretic Approach to Vulnerability Patching</i> , In International Conference on Information and Communication Technology Research (ICTRC) 88-91 (2015) (“Gianini”)
1041	Stefan Frei, <i>Security Econometrics: The Dynamics of (In)Security</i> , Ph.D. dissertation, ETH Zurich (2009) available at https://www.research-collection.ethz.ch/handle/20.500.11850/151394 (“Frei”)
1042	Mohit Kumar, <i>New Dark Web Marketplace Offers Zero-Day Exploits to Hackers</i> , The Hacker News (2015) (“Kumar”)
1043	<i>Famous Dark Net Marketplaces to but Exploits – 0 day Vulnerabilities – Malwares for Research</i> , International Institute of Cyber Security (2015) (“International Institute of Cyber Security”)
1044	Peter Mell, Tiffany Bergeron and David Henning, <i>Creating a Patch and Vulnerability Management Program</i> , NIST Special Publication 800-40, Version 2.0 (Nov. 2005) (“Mell”)
1045	Curriculum vitae (CV) of Dr. Erez Zadok.

31. I have considered these materials from the viewpoint of a POSITA as of the priority date of the '385 Patent (EX1001). For the purposes of this declaration, I have been asked to assume that the priority date of the '385 Patent is November 30, 2015. I note that my opinions provided in this declaration are made from the

perspective of a POSITA as of this priority date of the '385 Patent (EX1001) unless expressly stated otherwise. To the extent that I use any verb tense in this declaration that is present tense (*e.g.*, “a POSITA would understand” instead of “a POSITA would have understood”), such verb tense should be understood to be my opinion as of the '385 Patent's priority date (again, unless expressly stated otherwise). I merely use the present verb tense for ease of reading.

III. LEGAL FRAMEWORK

32. I am a technical expert and do not offer any legal opinions. However, counsel has informed me as to certain legal principles regarding patentability and related matters under United States patent law, which I have applied in performing my analysis and arriving at my technical opinions in this matter.

33. I have been informed that the Patent Trial and Appeal Board (“PTAB”) now applies the claim construction standard applied by Article III courts (*i.e.*, the Phillips standard) regardless of whether a patent has expired. I have been informed that under the Phillips standard, claim terms are to be given the meaning they would have to a person having ordinary skill in the art at the time of the invention, taking into consideration the patent, its file history, and, secondarily, any applicable extrinsic evidence (*e.g.*, dictionary definitions). I understand that, for purposes of this proceeding, Petitioner applies the ordinary and customary meaning of the claim terms as understood by a POSITA. I have also applied the plain meaning of the

claim language as instructed by Phillips.

34. I have also been informed that a person cannot obtain a patent on an invention if the differences between the invention and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art. I have been informed that a conclusion of obviousness may be founded upon more than a single item of prior art. I have been further informed that obviousness is determined by evaluating the following factors: (1) the scope and content of the prior art, (2) the differences between the prior art and the claim at issue, (3) the level of ordinary skill in the pertinent art, and (4) secondary considerations of non-obviousness. In addition, the obviousness inquiry should not be done in hindsight. Instead, the obviousness inquiry should be done through the eyes of a person of ordinary skill in the relevant art at the time the patent was filed.

35. In considering whether certain prior art renders a particular patent claim obvious, counsel has informed me that I can consider the scope and content of the prior art, including the fact that one of skill in the art would regularly look to the disclosures in patents, trade publications, journal articles, industry standards, product literature and documentation, texts describing competitive technologies, requests for comment published by standard setting organizations, and materials from industry conferences, as examples. I have been informed that for a prior art

reference to be proper for use in an obviousness analysis, the reference must be “analogous art” to the claimed invention. I have been informed that a reference is analogous art to the claimed invention if: (1) the reference is from the same field of endeavor as the claimed invention (even if it addresses a different problem); or (2) the reference is reasonably pertinent to the problem faced by the inventor (even if it is not in the same field of endeavor as the claimed invention). In order for a reference to be “reasonably pertinent” to the problem, it must logically have commended itself to an inventor’s attention in considering his problem. In determining whether a reference is reasonably pertinent, one should consider the problem faced by the inventor, as reflected either explicitly or implicitly, in the specification. I believe that all of the references that my opinions in this IPR are based upon are well within the range of references a person of ordinary skill in the art would consult to address the type of problems described in the Challenged Claims.

36. I have been informed that, in order to establish that a claimed invention was obvious based on a combination of prior art elements, a clear articulation of the reason(s) why a claimed invention would have been obvious must be provided. Specifically, I am informed that a combination of multiple items of prior art renders a patent claim obvious when there was an apparent reason for one of ordinary skill in the art, at the time of the invention, to combine the prior art, which can include, but is not limited to, any of the following rationales: (A) combining prior art methods

according to known methods to yield predictable results; (B) substituting one known element for another to obtain predictable results; (C) using a known technique to improve a similar device in the same way; (D) applying a known technique to a known device ready for improvement to yield predictable results; (E) trying a finite number of identified, predictable potential solutions, with a reasonable expectation of success; (F) identifying that known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art; or (G) identifying an explicit teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine the prior art references to arrive at the claimed invention.

37. I am informed that the existence of an explicit teaching, suggestion, or motivation to combine known elements of the prior art is a sufficient, but not a necessary, condition to a finding of obviousness. This so-called “teaching suggestion-motivation” test is not the exclusive test and is not to be applied rigidly in an obviousness analysis. In determining whether the subject matter of a patent claim is obvious, neither the particular motivation nor the avowed purpose of the patentee controls. Instead, the important consideration is the objective reach of the claim. In other words, if the claim extends to what is obvious, then the claim is invalid. I am further informed that the obviousness analysis often necessitates

consideration of the interrelated teachings of multiple patents, the effects of demands known to the technological community or present in the marketplace, and the background knowledge possessed by a person having ordinary skill in the art. All of these issues may be considered to determine whether there was an apparent reason to combine the known elements in the fashion claimed by the patent.

38. I also am informed that in conducting an obviousness analysis, a precise teaching directed to the specific subject matter of the Challenged Claim need not be sought out because it is appropriate to take account of the inferences and creative steps that a person of ordinary skill in the art would employ. The prior art considered can be directed to any need or problem known in the field of endeavor at the time of invention and can provide a reason for combining the elements of the prior art in the manner claimed. In other words, the prior art need not be directed towards solving the same specific problem as the problem addressed by the patent. Further, the individual prior art references themselves need not all be directed towards solving the same problem. I am informed that common sense is important and should be considered. Common sense teaches that familiar items may have obvious uses beyond their primary purposes.

39. I also am informed that the fact that a particular combination of prior art elements was “obvious to try” may indicate that the combination was obvious even if no one attempted the combination. If the combination was obvious to try

(regardless of whether it was actually tried) or leads to anticipated success, then it is likely the result of ordinary skill and common sense rather than innovation. I am further informed that in many fields it may be that there is little discussion of obvious techniques or combinations, and it often may be the case that market demand, rather than scientific literature or knowledge, will drive the design of an invention. I am informed that an invention that is a combination of prior art must do more than yield predictable results to be non-obvious.

40. I am informed that for a patent claim to be obvious, the claim must be obvious to a person of ordinary skill in the art at the time of the invention. I am informed that the factors to consider in determining the level of ordinary skill in the art include (1) the educational level and experience of people working in the field at the time the invention was made, (2) the types of problems faced in the art and the solutions found to those problems, and (3) the sophistication of the technology in the field.

41. I am informed that it is improper to combine references where the references teach away from their combination. I am informed that a reference may be said to teach away when a person of ordinary skill in the relevant art, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the patent applicant. In general, a reference will teach away if it suggests that the line of

development flowing from the reference's disclosure is unlikely to be productive of the result sought by the patentee. I am informed that a reference teaches away, for example, if (1) the combination would produce a seemingly inoperative device, or (2) the references leave the impression that the product would not have the property sought by the patentee. I also am informed, however, that a reference does not teach away if it merely expresses a general preference for an alternative invention but does not criticize, discredit, or otherwise discourage investigation into the invention claimed.

42. I am informed that even if a prima facie case of obviousness is established, the final determination of obviousness must also consider "secondary considerations" if presented. In most instances, the patentee raises these secondary considerations of non-obviousness. In that context, the patentee argues an invention would not have been obvious in view of these considerations, which include: (a) commercial success of a product due to the merits of the claimed invention; (b) a long-felt, but unsatisfied need for the invention; (c) failure of others to find the solution provided by the claimed invention; (d) deliberate copying of the invention by others; (e) unexpected results achieved by the invention; (f) praise of the invention by others skilled in the art; (g) lack of independent simultaneous invention within a comparatively short space of time; and (h) teaching away from the invention in the prior art.

43. I am further informed that secondary considerations evidence is only relevant if the offering party establishes a connection, or nexus, between the evidence and the claimed invention. The nexus cannot be based on prior art features. The establishment of a nexus is a question of fact. While I understand that the Patent Owner here has not offered any secondary considerations at this time, I will supplement my opinions in the event that the Patent Owner raises secondary considerations during the course of this proceeding.

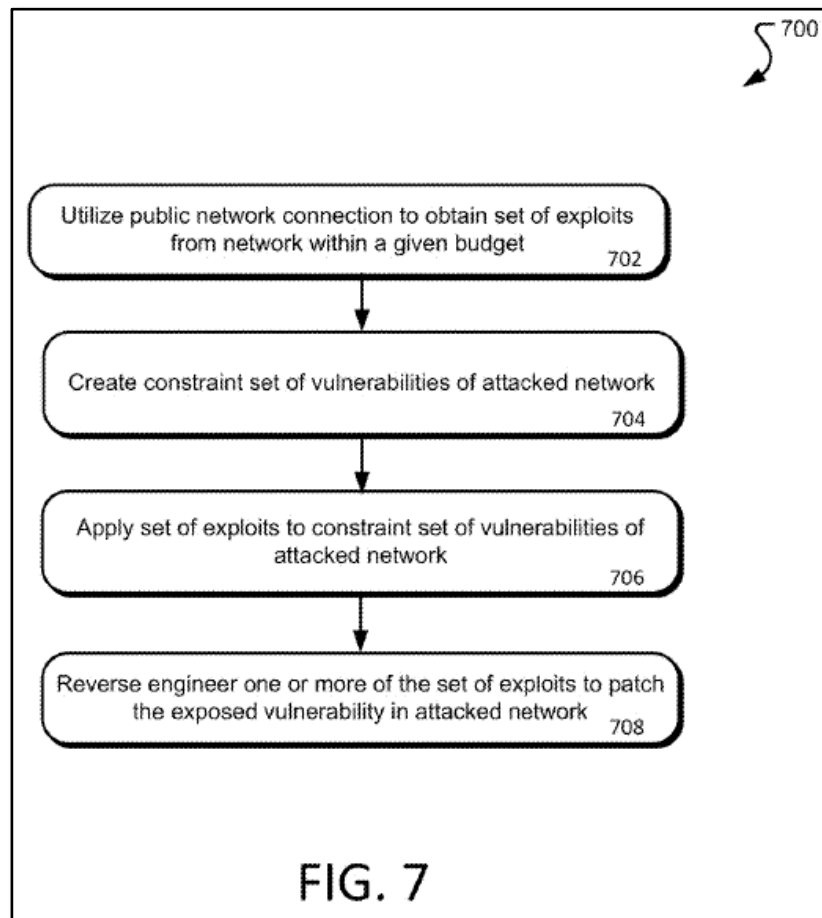
IV. OPINION

A. The '385 Patent

44. Generally directed to a framework for “addressing penetration testing” “fed with real-world exploit market data,” the '385 Patent purports to improve the “game theoretic approach to host-based defense where activities of the attacker are informed from an ‘un-conventional’ source ... specifically information from darknet markets[.]” '385 Patent (EX1001), 2:20-30, 3:4-6; '385 File History (EX1002), 125. The '385 Patent describes a game-theoretic model where an attacker purchases exploits to maximize her access to a target system, and a defender adjusts her targeted system to minimize the effect of the attack. '385 Patent (EX1001) 2:63-3:3. The model operates on a cost-optimization basis, requiring the attacker to purchase exploits within a given budget. *Id.*, 2:63-66, 4:7-8.

45. The claims broadly recite a typical penetration test using real-world

exploit information mined from the darknet. *Id.*, 2:30-31, 3:4-7. The claimed penetration test comprises: (1) accessing dark net information associated with exploits, (2) obtaining a set of exploits from that dark net information, (3) applying an exploit function to the set of exploits to identify a set of vulnerabilities, (4) creating a constraint set of vulnerabilities from the returned set of vulnerabilities, (5) applying the set of exploits to the constraint set, (6) analyzing the previous step to identify a particular vulnerability, and (7) patching the computer system based on the analysis to reduce the effect of a future attack. *Id.*, Claim 1; *see also id.*, 3:16-4:5, 9:5-33, 13:3-18. This test is depicted below in Figure 7:



Id., Fig. 7.

B. Summary of the '385 Patent's Prosecution History

46. The originally presented claims faced a single rejection. '385 Patent File History (EX1002), 87-93, 118-129. Original claim 1 is shown below for reference:

1. A system for protecting a computer system from attack, the system comprising:

- a network connection for communicating with a public network;
- a processing device; and
- a computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:
 - obtaining a set of exploits from the public network through the network connection, the set of exploits configured to penetrate the computer system;
 - creating a constraint set of vulnerabilities of the computer system comprising a minimum set of dependencies to operate the computer system;
 - applying the set of exploits to the computer system;
 - analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and
 - altering a configuration of the computer system in response to the analysis of the application of the set of exploits.

'385 Patent File History (EX1002), 37.

47. The Examiner found that the originally presented independent claims were obvious over U.S. Patent App. Pub. 2004/0088565 to Norman *et al.* ("Norman"), in view of U.S. Patent App. Pub. 2013/0198848 to Wolff ("Wolff" (EX1009)) and U.S. Patent App. Pub. 2014/0137257 to Martinez *et al.* ("Martinez"

(EX1010)). Specifically, the Examiner asserted that Norman teaches (1) “applying the set of exploits to the computer system”; (2) “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system”; and (3) “altering a configuration of the computer system in response to the analysis of the application of the set of exploits[.]” The Examiner asserted that Wolff teaches (4) “obtaining a set of exploits from public network through network connection[.]” and that Martinez teaches (5) “creating a constraint set of vulnerabilities of computer system comprising a minimum set of dependencies to operate the computer system[.]” ’385 Patent File History (EX1002), 89-90.

48. In response, Applicant amended the claims “to clarify that data is accessed through the dark net[.]” “to clarify that a set of exploits is obtained from the dark net information,” that “an exploit function is implemented to return a set of vulnerabilities[.]” and to “further clarify that the claimed subject matter is implemented to reduce potential damage of a cyberattack.” *Id.*, 126-128. An exemplary amendment of Claim 1 includes similar language:

1. (Original) A system for protecting a computer system from attack, the system comprising:

- a network connection ~~for communicating with a public network~~;
- a processing device; and
- a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:
 - accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:
 - obtaining a set of exploits from the dark net information ~~public network through the network connection~~, the set of exploits configured to penetrate the computer system;
 - applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;
 - creating a constraint set ~~of vulnerabilities~~ of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system;
 - applying the set of exploits to the computer system;
 - analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and
 - altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.

'385 Patent File History (EX1002), 119.

49. Prior to a Notice of Allowance, the Examiner entered an Examiner's Amendment to each of the independent claims. *Id.*, 137-139. These amendments are depicted below:

1. (Currently amended) A system for protecting a computer system from attack, the system comprising:

- a network connection;
- a processing device; and
- a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:

- accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:

- obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;

- applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;

- creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system;

- applying the set of exploits to the constraint set of the computer system;

Application/Control Number: 15/362,730
Art Unit: 2495

Page 3

- analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and
 - altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.

8. (Currently amended) A method for improving a computing device, the method comprising:
 accessing data comprising dark net information associated with a computer system;
 obtaining a set of exploits from the dark net information, the set of exploits configured to
 bypass a security feature of the computer system;
 applying an exploit function which takes the set of exploits as input and returns a set of
 vulnerabilities;
creating a constraint set of vulnerabilities of the computer system from the set of
 vulnerabilities comprising a minimum set of dependencies to operate the
 computer system, wherein application of the set of exploits on the computer
 system comprises determining the effect of the set of exploits on the constraint
 set of vulnerabilities of the computer system;
 analyzing an application associated with the set of exploits on the computer system to
 detect a particular vulnerability of the constraint set of vulnerabilities of the
 computer system; and
 altering a configuration of the computer system in response to the analysis of the
 application of the set of exploits to reduce potential damage of a cyberattack.

17. (Currently amended) One or more non-transitory tangible computer-readable storage media
 storing computer-executable instructions for performing a computer process on a machine, the
 computer process comprising:

Application/Control Number: 15/362,730
 Art Unit: 2495

Page 4

obtaining a set of exploits from dark net information, the set of exploits configured to
 bypass a security feature of ~~[[the]]~~ a computer system;
applying an exploit function which takes the set of exploits as input and returns a set of
 vulnerabilities;
creating a constraint set of vulnerabilities of the computer system from the set of
 vulnerabilities comprising a minimum set of dependencies to operate the computer system;
applying the set of exploits to the constraint set of vulnerabilities of the computer system;
 analyzing ~~[[an]]~~ the application of the set of exploits on the computer system to detect a
 particular vulnerability of the computer system; and
 altering a configuration of the computer system in response to the analysis of the
 application of the set of exploits to reduce potential damage of a cyberattack.

Id.

50. A Notice of Allowance issued thereafter. *Id.*, 131-141.

51. As explained in the Grounds below, Vallone (EX1004), Oberheide (EX1005), Patsos (EX1006), Roytman (EX1007), Juels (EX1008), and Wolff (EX1009) disclose the allegedly patentable system and methods for using exploit data mined from the dark net to exploit vulnerabilities in a computer system and then remediate those vulnerabilities. *See* §VI below, §VII below, §VIII below, and §IX below.

C. Claim Construction

52. I have been informed by counsel that the first step in an unpatentability analysis involves construing the claims, as necessary, to determine their scope. Second, the construed claim language is then compared to the disclosures of the prior art. I am informed that claims are generally given their ordinary and customary meaning as understood by one of ordinary skill in the art at the time of the invention in light of the patent specification and prosecution history, *i.e.*, the intrinsic record. For purposes of my analysis, I have been asked to apply the plain and ordinary meaning as defined in this paragraph.

D. Level of a Person of Ordinary Skill in the Art

53. I understand that the '385 Patent (EX1001), the record of proceedings at the Patent Office (which I understand is called the "File History" or "Prosecution History"), and the teachings of the prior art, are evaluated from the perspective of a person of ordinary skill in the art ("POSITA") at the time the claimed invention was

made. I understand that the factors considered in determining the ordinary level of skill in the art include: (i) the levels of education and experience of persons working in the field; (ii) the types of problems encountered in the field; and (iii) the sophistication of the technology. I may also consider, if available, prior art solutions to the problems encountered in the art and the rapidity with which innovations are made in the relevant art.

54. I understand that a person of ordinary skill in the art is not a specific real individual, but rather a hypothetical individual having the qualities reflected by the factors above. This hypothetical person has knowledge of all prior art in the relevant field as if it were arranged on a workshop wall and takes from each reference what it would teach to a person having the skills of a POSITA.

55. Based on my review and analysis of the '385 Patent (EX1001), the prior art cited herein, and the ordinary skill factors described in this section, a POSITA in the field of the '385 Patent at the time of the earliest possible priority date (Nov. 30, 2015) would have had a bachelor's degree in computer science, electrical or computer engineering, or an equivalent degree; and 2-3 years of work or research experience in computer security and networking. Additional education (*e.g.*, MS degree) might substitute for less experience and vice versa.

56. As of Nov. 30, 2015, I had more than ordinary skill in the art and meet at least these minimum qualifications to be a person having ordinary skill in the art

as of the time of the claimed invention of the '385 Patent. I am, however, familiar with the skills and knowledge possessed by those I would have considered to be of ordinary skill in the art as of that date. *See* §I above, Educational Background and Qualifications.

57. My opinions provided in this declaration would not change in view of minor modifications to this level of ordinary skill.

V. BACKGROUND OF THE TECHNOLOGY

58. The features described and claimed in the '385 Patent were routine as of November 30, 2015. A 2008 publication from the National Institute of Standards and Technology (NIST) offers official guidance from the U.S. government on how organizations should conduct security tests. *See generally*, Scarfone (EX1011). The publication includes discussion on penetration testing to identify and target vulnerabilities, and its broad phases of planning, discovery, attack, and reporting. *Id.*, 5-2–5-6.

59. The application of this security assessment in a game-theoretic model was also common before the filing of the '385 Patent. Game theory itself is a long-understood concept dating back to the early 20th century. Microsoft (EX1012), 201 (defining game theory and attributing it to 1900's mathematician Jon von Neumann). A 2010 paper discusses "A Game-Theoretic Framework for Network Security Assessment": it considers the actions of both an attacker and defender, including a

cost/benefit tradeoff associated with the defender's investment in security. Gueye (EX1013), 1-2 (describing effective risk management as including identifying an optimal operating point "given specific user security and cost preferences"). Prior still, security assessments have included a focus on "tradeoffs in terms of *investment costs* and *opportunity cost* (from perspective of defender and attacker respectively). Tiwari (EX1014), 1-2 (explaining that an attacker needs the payoff from an exploitation to outweigh the cost of the exploitation itself).

60. Next, I briefly summarize the technology background for a few particular routine features of the '385 Patent that are relevant to the opinions set forth herein.

A. Web Crawling Using Algorithms

61. Web crawlers are common tools for systematically navigating and extracting data from the Internet. Olston (EX1015), for example, describes crawlers as featuring prominently in applications involving the collection of web data, such as web archiving, web monitoring, and data mining. Olston, 176. "A web crawler is a program/software or programmed script that browses the World Wide Web in a systematic, automated manner." Kauser (EX1016), 31; *see also* Mirtaheri (EX1017), 1 ("Crawling is the process of exploring web applications automatically.").

62. As a result of their automatic character, web crawlers are also referred to as “a robot[.]” Olston (EX1015), 176. The automatic/robotic of web crawling, regardless of the application, is due to a “web crawling algorithm” that follows this basic formula: “Given a set of seed Uniform Resource Locators (URLs), a crawler downloads all the web pages addressed by the URLs, extracts the hyperlinks contained in the pages, and iteratively downloads the web pages addressed by these hyperlinks.” Olston (EX1015), 178; *see also* Mirtaheri (EX1017), 1 (describing a web crawler as “basically a software” that operates through the same iterative process).

63. Web crawling was known long before the priority date of the '385 Patent. *See* Olston (EX1015), 180 (“Web crawlers are almost as old as the web itself.”). Similarly, crawlers were known to be directed for scanning and collecting information from the dark web. Olston, 230 (describing “deep web crawlers”); Fu (EX1018) (emphasizing crawling the dark web). More pointedly, web crawlers have also been used for more specific tasks, including “work on crawling discussion forums[.]” Olston (EX1015), 237. Research including Yang (EX1019), 1375 (discussing a web strategy for “forum crawling”); and Fu (EX1018), 1213 (proposing “a focused crawler that can collect Dark Web forums”) is evidence of this work. Therefore, a web crawler was understood as an algorithm-implemented

tool capable of obtaining data from web pages at large, including the dark web (*e.g.*, dark web forums).

64. Indeed, it was a then small company named Google who popularized massive, automated, algorithmic web crawling and indexing in the late 1990s, which I observed personally. Those crawling and indexing efforts have been so successful, that well before 2015 a new verb emerged—“to google” something.

B. Retrieving Exploit Data from Dark Net Markets

65. It was well-known before the priority date of the '385 Patent that penetration testers could use exploits obtained from the dark net markets. Penetration testing (or “pentesting”) operates by using “real-world attacks” on a target system. Scarfone (EX1011), 5-3; *see also* Obes (EX1020), 1 (“pentesting frameworks have the ability to launch real exploits for vulnerabilities, helping to expose risk by conducting an attack in the same way a real external attacker would”).

66. Dark net markets are a common source for accessing and obtaining these “real” exploits. Miller (EX1021), 2 (“There has long been a black market for computer exploits.”); *see also* Chertoff (EX1022), 5 (citing “Exploit Markets” among the types of cybercrime present on the dark web). Though much of the activity across the dark net is criminal in nature, the exploit market does include at least some legitimate business uses. Chertoff (EX1022), 4-5; Miller (EX1021), 2 (“A black market for these exploits has developed around their illegal use. Recently,

a trend has developed toward buying and selling these exploits as a source of legitimate income for security researchers.” [...] “Such legal buyers may include [...] large penetration testing and consulting firms”).

67. It should be noted that there is nothing special about “dark nets” in the context of this patent. Harvesting dark net data may benefit from using custom crawlers, not claimed in this patent. The same techniques used to analyze publicly available information (*e.g.*, text, html, source code, blog posts, etc.) can be applied with little (if any) changes to data gathered from dark net sources. While modern dark nets may use high-speed networks, the web, and Internet—in the late 1980s and 1990s such “dark” information was usually posted on obscure USENET newsgroups as well as disseminated via Bulletin Board Systems (BBSs) one had to connect via slower dial-up modems. Lastly, while some entities may try to sell such exploits for illicit purposes, it is often a matter of cost levels, because using computer networks, storage, etc. always costs something (*e.g.*, renting space on clouds, paying for network connections, energy and cooling costs of computers, etc.).

C. Identifying Vulnerabilities Relevant to the Mined Exploits

68. Prior to November 30, 2015, it was a known practice to correlate the obtained exploits discussed above to the vulnerabilities they are capable of exploiting. The NIST publication presents a variety of assessments, including some that “are intended to identify, validate, and assess a system’s exploitable security

weaknesses[,]” *i.e.*, vulnerabilities. Scarfone (EX1011), ES-1.

69. As an overall process, a security assessment comprises at least planning, execution, and post-execution phases. The planning phase focuses on information gathering in preparation for the execution phase, including information on “the threats of interest against the assets” of the computer system. Scarfone (EX1011), 2-2. Naturally, the “threats of interest” are the exploits. The execution phase follows by using the information gathered in planning to perform operations comprising the identification of vulnerabilities. *Id.* (“Primary goals for the execution phase are to identify vulnerabilities and validate them when appropriate.”).

70. Penetration testing is one security testing technique that may perform the above steps, which Scarfone describes as part of a “discovery” phase of the penetration test. *Id.*, 5-3 (describing discovery as including two steps: one for information gathering and a second for vulnerability identification/analysis). Thus, the identification of system vulnerabilities results from analysis of information previously gathered, that information including exploits that are a threat to a target computer system. In the ’385 Patent (EX1001) and as discussed in §V.B above the exploits used for identifying vulnerabilities may have been gathered from dark net markets.

D. Applying Exploits to Determine Attack Patterns

71. Attack paths were known as a means for correlating exploits to

vulnerabilities before the priority date of the '385 Patent. It is well known that attack paths exploit certain vulnerabilities that allow attackers to achieve their objective. Carvalho (EX1023), 2 (“Network attack graphs are directed graph-representations of possible attack paths and vulnerabilities in a computer network. Each attack path is a sequence of steps taken by an attacker to achieve one or more goals in the target system.”).

72. An attack path may be built by correlating a set of exploits to the conditions required for the exploits to be effective. In Jajodia (EX1024), for example, a Topological Vulnerability Analysis tool is capable of analyzing attack paths. Jajodia (EX1024), 1. An attack graph is built from correlating exploits and vulnerabilities. Particularly, Jajodia models each exploit as “a rule in which the occurrence of a particular set of preconditions induces a particular set of postconditions.” *Id.*, 5; *see also* Carvalho (EX1023), 4 (each “attack step is defined by its pre- and postconditions”).

73. The pre-conditions of an attack step are the prerequisites that must be satisfied for the attack step to be successful. Carvalho (EX1023), 4. The post-conditions are the consequences that will result once this attack step has been executed. *Id.* The pre- and post-conditions include *vulnerabilities* and other network security conditions. Jajodia (EX1024), 2. Thus, each exploit rule is correlated to the gathered vulnerability information (the pre- and post-conditions) upon which it

depends. *Id.* “The result is a set of attack paths leading from the initial network state to a pre-determined attack goal.” *Id.*

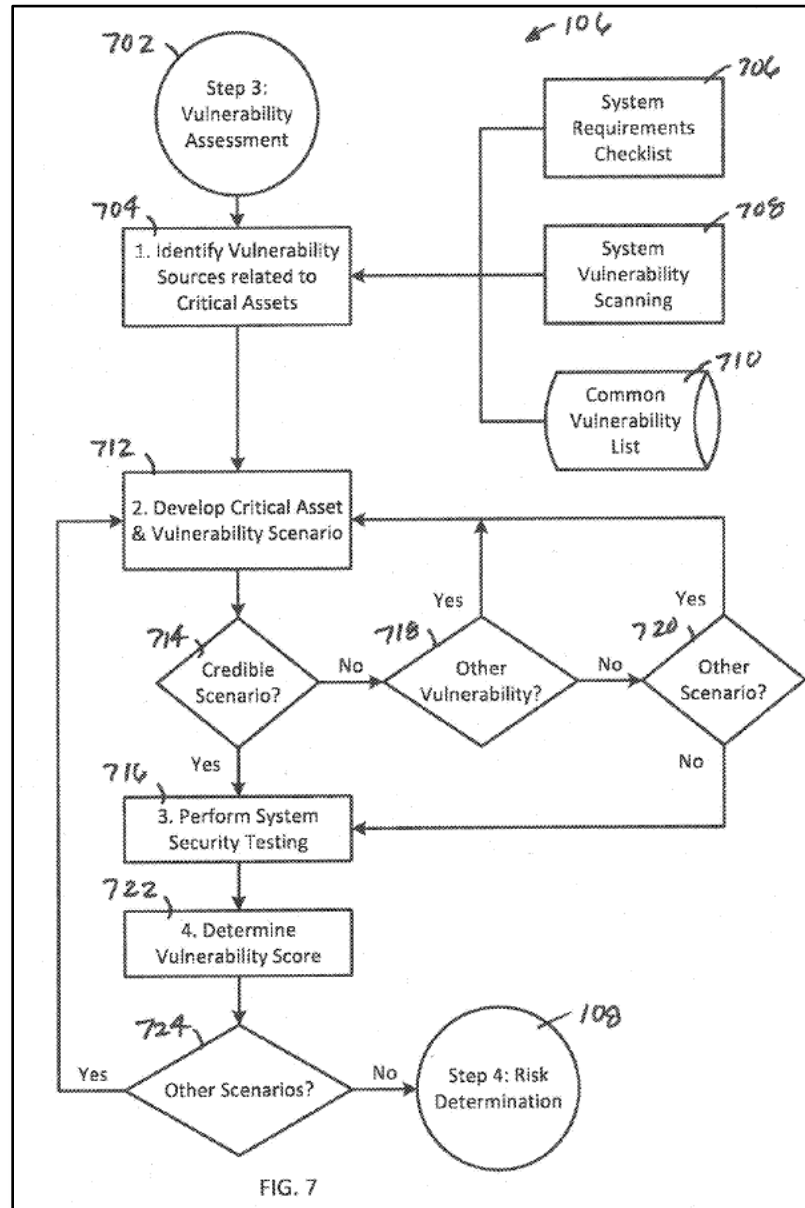
E. Identifying a Limited Group of Vulnerabilities to Test

74. Before the filing of the alleged invention, it was known that security testing could be performed on a subset of those vulnerabilities identified for the system as a whole. One benefit of this identification is to narrow the focus of testing on the most vulnerable portions of the most essential system components.

75. As one example, Martinez (EX1010) describes an infrastructure that contains electronic components with a number of vulnerabilities at risk of exploitation. *Id.*, [0048]. Some of the electronic components support “critical assets” (CAs), defined as “physical component[s] essential to the operation of the infrastructure.” *Id.* A vulnerability assessment step of *Martinez* includes identifying a subset of those vulnerabilities that are related to the critical assets in the infrastructure. *Id.*, [0109] (“The goal of this step is to develop a list of critical asset vulnerabilities that could be exploited by the potential threat-sources.”).

76. System security testing, including penetrating testing, is then performed based on the pairings of the critical assets and their vulnerabilities. *Id.*, [0116]. Here, Martinez discusses a step-by-step process for vulnerability assessment, which includes “[v]ulnerability sources related to critical assets are identified in block 704 using a system requirement checklist 706, system vulnerability scanning 708, and/or

common vulnerability list 710. A critical asset and vulnerability scenario is developed in block 712. If the scenario is credible, as determined in decision block 714, system security testing is performed in block 716.” *Id.*; *see also, id.*, Fig. 7 below:



F. Testing Exploits in a Protected Environment

77. Prior to the alleged invention, it was known that exploit testing could be carried out in isolated, controlled environments. Brickell (EX1025), Abstract (“Improving security of a processing system may be accomplished by at least one of executing and accessing a suspect file in a sandbox virtual machine.”). In a “sandbox” or similar virtual environment, a malicious code may execute an attack just as it would in a typical computer system. Because the attack occurs in a controlled synthetic environment, however, “the code may be quarantined so that the code cannot damage or access any of the computing platform outside of the sandbox.” Brickell (EX1025), [0029]; *see also* Miwa (EX1026), 1 (defining “isolated sandbox” as a mechanism designed to avoid any impacts to the outside from the analysis in an isolated analyzing environment).

78. “Honeypots” provide a similar benefit. For example, Nicomette (EX1027) describes deploying a honeypot “1) to provide to the attackers a vulnerable target operating system, and 2) to ensure that the potential risks that might result from the exploitation of such vulnerabilities are properly controlled and mastered.” Nicomette (EX1027), 6.

79. Whether employing a sandbox, honeypot, etc., this concept enables testers to observe exploit behavior and identify/address issues in a computer system based on those observations. Miwa (EX1026), 1 (stating that “[i]solated sandboxes

are good analyzing environments for carrying out ... observations [of malware threat behavior] because they have tolerance to attacks and infections from the outside.”). Thus, creating an isolated environment was understood by the cybersecurity community at large as an option for executing and testing exploits.

G. Remediating the Computer System Based on the Results of Testing

80. It was well understood before the priority date of the '385 Patent that penetration testing may be followed by measures to eliminate or mitigate the exploited vulnerabilities. The motivation to employ patching mechanisms is improved security. Scarfone (EX1011), 8-1 (describing remediation and mitigation as “actions that will improve security.”); *see also* Martinez (EX1010), [0129] (discussing how to identify and evaluate “risk management strategies” whereby “[i]f the identified risk management strategy lowers risk ... a report document is prepared[,]” and if “the risk level is not high, ... general security countermeasures are applied).

81. The NIST Special Publication 800-115 provides an overview of recommended post-testing activities such as remediation and mitigation. Scarfone (EX1011), 8-1–8-3 (“Following the execution phase—whose findings are expressed in terms of vulnerabilities—the organization should take steps to address the vulnerabilities that have been identified.”). Scarfone breaks the post-execution phase into the steps of developing mitigating/remediation recommendations,

reporting those recommendations, and finally carrying out the recommendations. *Id.*, 8-1. For example, a recommendation of “applying a particular patch” may include installing the patch on the targeted computer system to reduce or eliminate the exploitable vulnerabilities. *Id.*, 8-2.

82. Installing patches is not the only means of mitigating vulnerabilities. Among other options, remediation may alternatively be achieved by uninstalling or otherwise removing the vulnerable software component. Oliphant (EX1028), [0033] (teaching remediation techniques of device vulnerabilities including “installation of a patch [...]; changing the device’s configuration; ***stopping, disabling, or removing services***,” and more)¹. Removing vulnerable services or software is beneficial in that it “reduc[es] risk to organizations by giving attackers less opportunity to exploit weaknesses or deficiencies” within computer systems. Joint Task Force (EX1029), F-175, F-168 (weaknesses “includ[e] the hardware, software, and firmware components”); F-175–76 (reducing an attack surface by restricting services and “eliminating application programming interfaces (APIs) that are vulnerable to cyber attacks”).

¹ All bold and bold-italicized emphases in this Declaration are added unless otherwise noted.

VI. OPINION – GROUND 1

A. Obviousness over Vallone

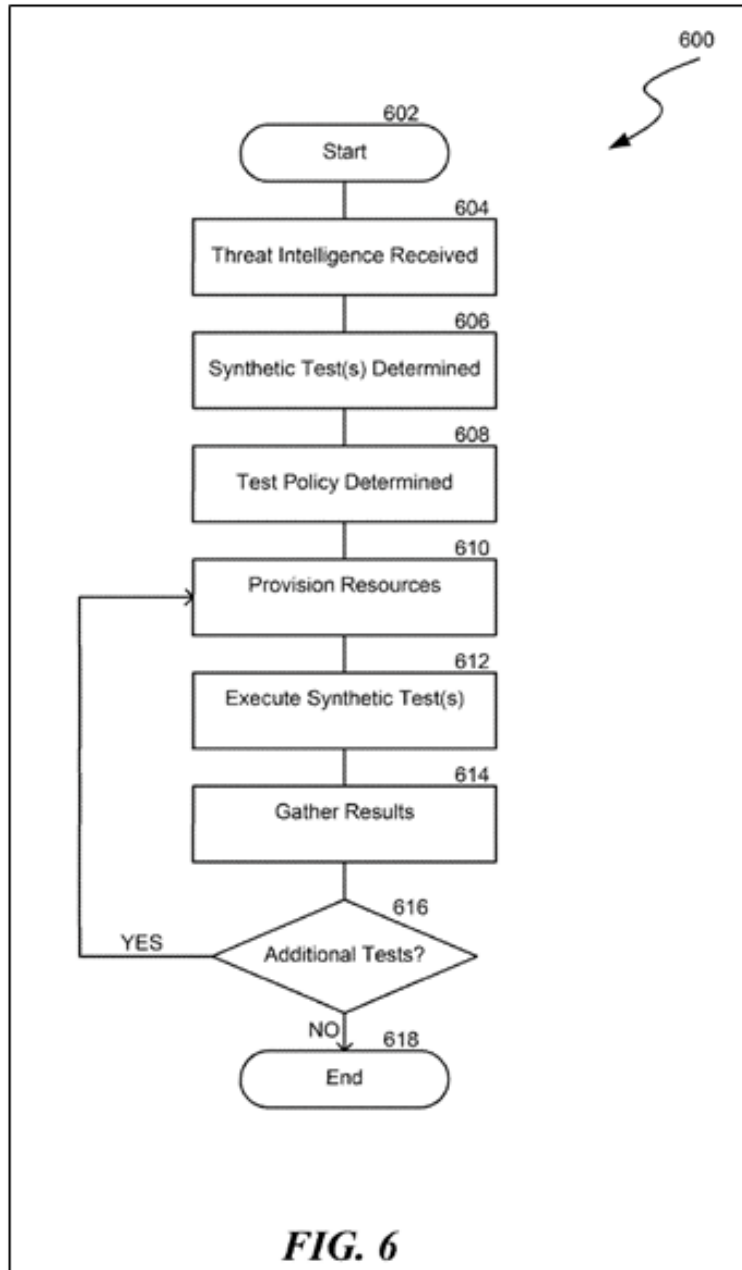
83. Claims 1, 8, 14, and 15 would have been obvious over Vallone (EX1004).

1. Overview of Vallone

84. U.S. Patent Publication No. 2016/0134653 to Vallone *et al.* (“Vallone”) was filed on November 11, 2014, and published on May 12, 2016 (EX1004, cover).

85. Vallone teaches a “device ... for assessing a target network’s vulnerability to a real cyberthreat [using] synthetic tests configured to model the behavior of the cyberthreat.” Vallone (EX1004), Abstract. Vallone’s device “access[es] cyberthreat intelligence networks ... [such as] **the dark net**, etc.” to determine cyberthreat objectives, obtains “threat characteristics” including the cyberthreat’s sequence of actions, executes instructions at a host computer “with the intent of generating the threat’s characteristics into the target environment,” “monitors target network 300 activity during [the] synthetic test,” and provides “feedback data” to “increase the efficiency at which [the] target network 300 can heal any vulnerabilities to one or more cyberthreats.” *Id.*, Abstract, [0016], [0019], [0039], [0040].

86. This method is illustrated in Figure 6:



Id., Fig. 6.

87. Vallone’s synthetic penetration test is performed by “agents” installed on “host” computers for testing the vulnerabilities on the target network. *Id.*, [0015]-[0017]. In this way, Vallone’s system is “able to synthesize the cyberthreat ... so that the ultimate objective of the cyberthreat ... is not committed, while still

providing an administrator with knowledge ... regarding the depth of vulnerability of the target network 300 to an actual attack.” *Id.*, [0038].

88. Because Vallone and the ’385 Patent are both directed to systems and methods for exploiting vulnerabilities and patching vulnerabilities, Vallone is in the same field of endeavor as the ’385 Patent. *Compare* ’385 Patent (EX1001), 3:4-15, 3:61-64, 13:12-18, Figs. 5 and 7 *with* Vallone (EX1004), Abstract, [0021]-[0022], [0038]-[0040], Fig. 6.

89. For example, the ’385 Patent describes common problems in the art related to developing patches for system vulnerabilities. ’385 Patent (EX1001), 2:63-3:3 (describing a defender looking “to adjust system configurations in an effort to minimize the effectiveness of an attacker”), 13:15-18 (“the computing device may reverse engineer one or more of the set of exploits applied to the attacked network to develop a patch to the exposed vulnerability”).

90. Vallone similarly addresses the problem of developing patches for system vulnerabilities. Vallone (EX1004), Abstract (disclosing “self-healing processes that, using the feedback mechanisms can attempt to find patches for known vulnerabilities”), [0020]-[0022] (“the technology can dynamically update the target network’s security capabilities to, e.g., protect against the same and/or another vulnerability ... by downloading and/or installing updates, firmware, new releases, antivirus rules, etc.”). As such, Vallone is reasonably pertinent to the problems

addressed by the '385 Patent and is analogous art.

2. *Claim 1(Pre): “A system for protecting a computer system from attack, the system comprising:”*

91. As an initial matter, I note that claim 1 is not limited to the claimed “computer system” itself as executing all the claimed functions below.

92. To the extent the preamble is limiting, Vallone teaches systems for “modeling risk of cyber-attacks[.]” Vallone (EX1004), [0001]. Vallone’s system “assess[es] a target network’s vulnerability to a real cyberthreat” and uses feedback “to find patches for known vulnerabilities[.]” *Id.*, Abstract. This process is used to “dynamically update the target network’s security capabilities to, e.g., protect against the same and/or another vulnerability.” *Id.*, [0020]-[0021]. A POSITA would have understood that patching vulnerabilities protects a computer system from attack.

93. Vallone’s system is directed to protecting computer systems within a network from attack. For example, Vallone discloses updating “servers 310 and hosts 320 with recent security updates[.]” *Id.*, [0040]. By updating both servers and hosts, Vallone protects computer systems (*i.e.*, “hosts”) by enhancing security at the network level (*e.g.*, by updating servers of the network) and directly at the computer system level (*e.g.*, by updating hosts). In this way, Vallone provides multiple methods of protecting computer systems from attack.

94. **In sum**², Vallone teaches “[a] system for protecting a computer system [Vallone’s host computer] from attack [Vallone’s patching host computers].”

3. Claim 1(a): “a network connection;”

95. Vallone’s system is connected to a network. Vallone (EX1004), [0037]. Specifically, “[t]hreat simulation technology 500 is configured to communicate to **external network 250** to ... receive information regarding cyberthreats” by “**access[ing] cyberthreat intelligence networks**, local and/or remote data repositories (e.g., data 520), the dark net, etc. to determine whether a new cyberthreat has been released[.]” *Id.* [0037], [0040]; *see also* Fig. 2B.

96. **In sum**, Vallone teaches “a network connection [Vallone’s external network 250].”

4. Claim 1(b): “a processing device; and”

97. Vallone’s system includes a processing device. Vallone (EX1004), [0028]. Vallone teaches its system includes “routines executed by a general- or special-purpose data processing device (e.g., a server or client computer).” *Id.* Vallone’s “threat simulation technology ... includes data 520 and server(s) 510[.]”

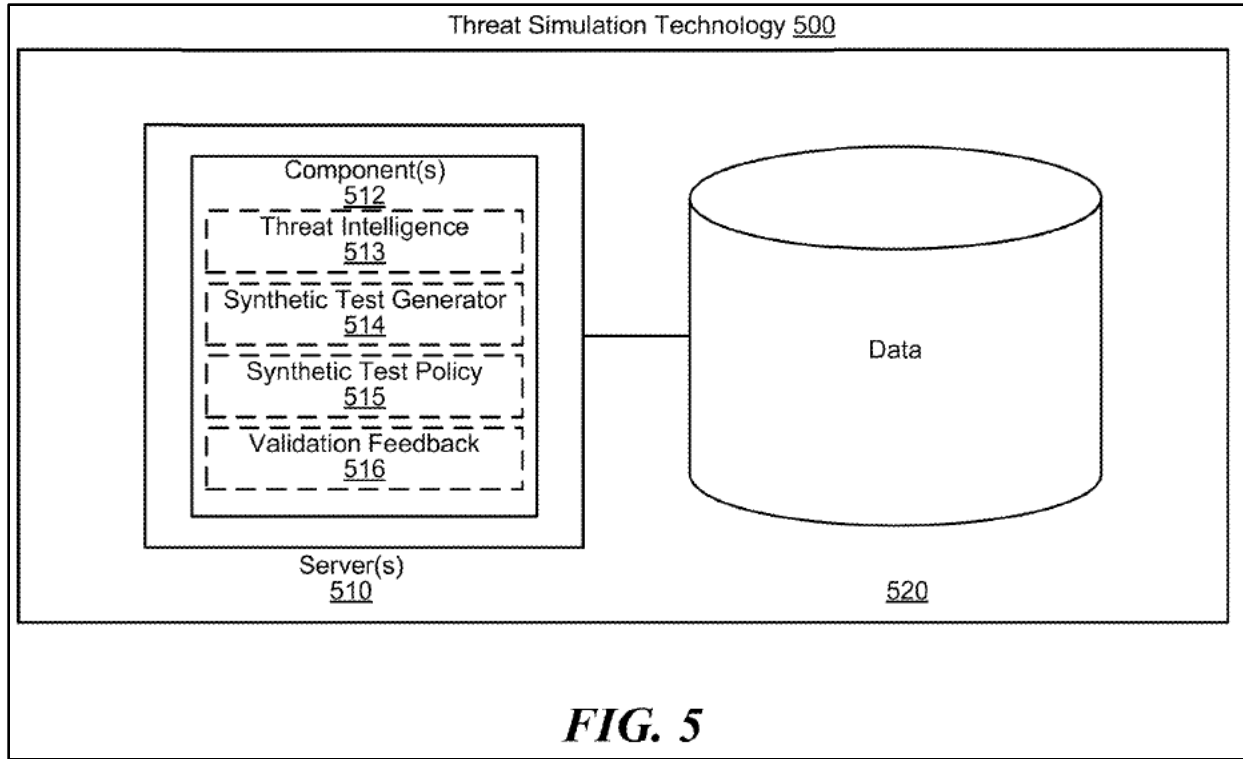
² Each summary paragraph is exemplary and overviews an example of how the prior art teaches and/or renders obvious the claims. My analysis is not limited to these examples and encompasses the other examples described herein.

Id., [0040], Figs. 2B, 5. The server (*i.e.*, processing device) of Vallone’s threat simulation technology performs the claimed steps, as described in §§VI.A.6-VI.A.28.

98. **In sum**, Vallone teaches “a processing device [Vallone’s server 510].”

5. ***Claim 1(c): “a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:”***

99. Vallone teaches “computer executable instructions” “stored on a machine-readable storage medium [that are] executed by one or more general-purpose or special-purpose programmable microprocessors.” Vallone (EX1004), [0028], [0045]. Vallone provides example “non-transitory” machine-readable storage media, including ROM, RAM, magnetic disc storage, optical storage, and flash memory devices. *Id.* With respect to Vallone’s specific embodiments, Vallone’s threat simulation technology includes “data 520” connected to “server 510[.]” Vallone (EX1004), Figs. 2B, 5 (below):



Id., Fig. 5.

100. Vallone’s ROM with data 520 is “a non-transitory computer-readable medium connected to the processing device [*i.e.*, server 510] configured to store instructions” that are executed by server 510.

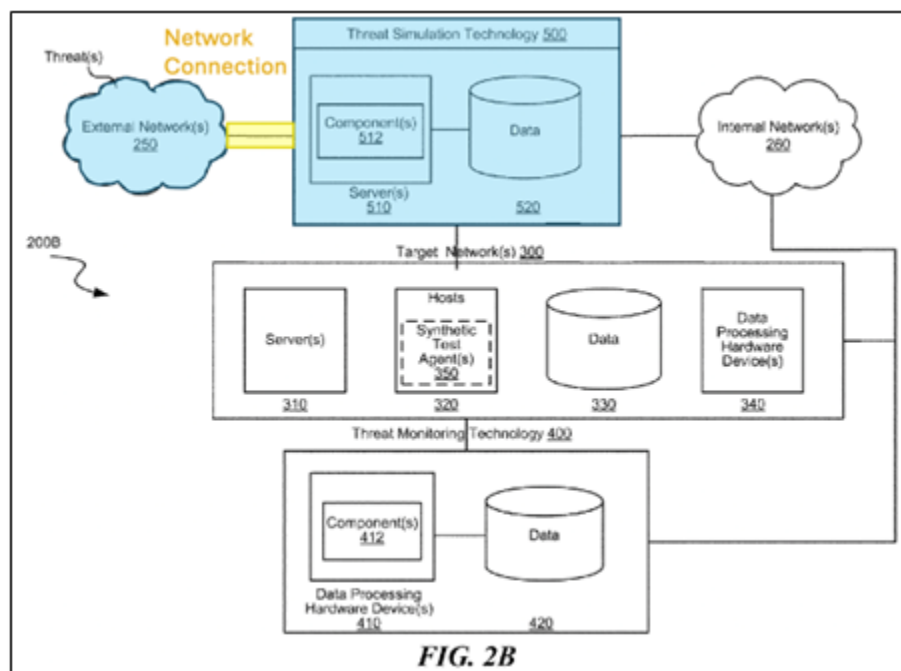
101. **In sum**, Vallone teaches “a non-transitory computer readable medium [Vallone’s ROM] connected to the processing device [Vallone’s server 510] configured to store instructions that, when executed by the processing device [Vallone’s server 510], performs the [claimed] operations[.]”

6. ***Claim 1(d): “accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:”***

102. As an initial matter, a POSITA would have recognized that there is

nothing special about this limitation's use of "dark net" information because the claim would operate the same on public, non-dark net data. *See* §V.B above.

103. Vallone teaches its threat simulation technology "fetch[es] ... information regarding the existence and details regarding cyberthreats." Vallone (EX1004), [0040]. "Cyberthreats" include "viruses, Trojans, rootkits, **zero-day exploits[.]**" *Id.*, [0005]. Vallone teaches that "threat intelligence component 513 can access" "the **dark net** [] to determine whether a new cyberthreat has been released and, if so, what its capabilities and objectives are[.]" *Id.*, [0040]. Threat simulation technology "receive[s] information regarding [dark net] cyberthreats" over network connection 250. *Id.*, [0037]. This process is depicted below in Vallone's annotated Figure 2B:



Id., Fig. 2B (annotated).

104. **In sum**, Vallone teaches “accessing data [Vallone’s fetch[ing]] through the network connection [Vallone’s network connection between threat simulation technology 500 and external network(s) 250], the data comprising dark net information associated with exploits [Vallone’s dark net cyberthreats] of a computer system [Vallone hosts 320].”

7. ***Claim 1(e): “obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;”***

105. As an initial matter, a POSITA would have recognized that there is nothing special about this limitation’s use of “dark net” information because the claim would operate the same on public, non-dark net data. *See* §V.B above.

106. I have been informed that Skysong accuses CrowdStrike’s Falcon Platform and specifically the Falcon Adversary Intelligence module of infringing this limitation. Infringement Contentions (EX1030), 45-46 (further accusing older modules including “Falcon Platform Falcon Intelligence” and “False Intelligence Recon+”).³ To “obtain[] a set of exploits from the dark net information,” I understand that Skysong alleges that these modules “implement ‘dark web

³ I have been asked to apply Skysong’s interpretation of the claim language outlined in its infringement contentions to my opinions in Ground 1. I have no opinion as to the correctness of Skysong’s claim interpretations.

monitoring’ to ‘develop a more comprehensive understanding of attackers and their methods’.” *Id.*, 45 (citing CrowdStrike’s website that states “[d]ark web monitoring continuously searches the dark web and pulls in raw intelligence in near real time ... The data captured by the dark web monitoring solution can be fed into automated threat intelligence systems and used to enrich that data”).

107. For application of alleged exploit code, I understand that Skysong accuses the CrowdStrike Falcon Sandbox that “‘establishes a space where potentially harmful code can be safely executed’ ... in which ‘investigators can ensure the rest of the system remains unaffected, thereby minimizing the risk of a security breach’.” *Id.*, 53-60. That is, Skysong alleges that CrowdStrike’s Falcon Platform obtains exploit code/script from the dark web and applies it in a sandboxed environment. To the extent the claims are interpreted such that “obtaining a set of exploits” includes crawling the dark net for exploit code, Vallone teaches this limitation.

108. Vallone teaches “obtain[ing]” “a threat characteristic” which “is a threat’s signature, action, sequence of actions or other consequence of the real threat being executed.” Vallone (EX1004), [0016]. These “[c]haracteristics are mapped to instructions via logic ... that when executed by an agent would achieve ... an objective of the cyberthreat.” *Id.*, [0017]. For example, Vallone receives threat intelligence from, *e.g.*, the dark net, “defin[ing] a series of instructions (*e.g.*, a script

or program) that, when carried out by one or more agents, will exhibit the malware's threat characteristics." *Id.*, [0019], [0038].

109. Specifically, Valone discloses that "the threat intelligence component 513 can access cyberthreat intelligence networks, local and/or remote data repositories (e.g., data 520), **the dark net**, etc. to determine whether a new cyberthreat has been released and, if so, what its capabilities and objectives are...." *Id.*, [0040]. By obtaining threat characteristics regarding multiple cyberthreats from the dark net, Vallone's system "obtain[s] a set of exploits from the dark net information[.]" *See also* §V.B above (discussing obtaining data from dark net markets was commonplace).

110. More specifically, Vallone's scripts or programs configured to execute the series of steps of a malware attack would have been understood as exploits because they behave in the same way as an exploit. For example, these scripts would have exploited one or more vulnerabilities (e.g., in an attack path) to gain access to computer components that would otherwise be protected. Thus, to the extent that obtaining exploit code/script from the dark web as accused includes "obtaining a set of exploits," the script or programs obtained by Vallone's system are likewise "exploits" as claimed.

111. These exploits are "configured to penetrate the computer system[.]" Specifically, Vallone's threat simulation technology includes a synthetic test

generator that “determines, for at least a portion of a cyberthreat’s objective, one or more steps of an instruction for emulating that ... objective.” Vallone (EX1004), [0040]. These objectives may penetrate a computer system to, *e.g.*, “steal usernames and passwords[.]” *Id.* (teaching determining multiple steps for an exploit, including “running a Flash browser-based script ‘badscript.fl’ and creating a tunnel between data 520 and an Internet server” to “infiltrat[e] a network”); *see also, id.*, [0016] (teaching threat characteristics, including “generating a file, creating a network communication, accessing a user account/directory/database/registry, a sequence of actions performed by the threat etc.”).

112. A POSITA would have understood that the exploits Vallone obtains and tests on its system are configured to penetrate the system. As explicitly taught by Vallone (EX1004), consequences of a cyberthreat would all require “penetration” of the system (“*e.g.*, theft of data, willful corruption of data, deletion/redirection/injection of data”). *Id.*, [0016]. To complete any of these objectives, the exploit must gain unprivileged access (*i.e.*, penetrate) the computer system. As such, the exploit sets obtained in Vallone are “configured to penetrate the computer system.”

113. **In sum**, Vallone teaches “obtaining a set of exploits [Vallone’s script or programs that exhibit a malware’s threat characteristics (*e.g.*, the threat’s signature, action, or sequence of actions)] from the dark net information, the set of

exploits configured to penetrate the computer system [Vallone's scripts are configured to hijack one or more portions of the computer system to perform unprivileged tasks]”.

8. ***Claim 1(f): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”***

114. As an initial matter, a POSITA would have recognized that the term “function” in the ’385 Patent is not limited to a mathematical function such that it formulaically takes an input and returns an output, *e.g.*, $y = f(x_1, x_2, \dots)$. For example, a POSITA would have understood, based on the intrinsic record, that there are “many possible variations of an exploit function [that] are possible[.]” ’385 Patent, 3:64-4:5.

115. Vallone teaches applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities by determining the “sequence of actions” an exploit takes to penetrate the computer system. Vallone (EX1004), [0016], [0040]. Vallone’s synthetic test generator inputs exploits and determines “one or more steps of instructions” for emulating the exploits in synthetic tests. Vallone (EX1004), [0041]; *see also, id.* (“synthetic tests are determined by **mapping the exploit’s characteristics (e.g., any files created/accessed, communications spawned) into one or more steps of instructions**”). *See also* §V.C above V.C (discussing how linking vulnerabilities to mined exploits was commonplace).

116. For example, Vallone teaches obtaining sets of cyberthreats (*i.e.*,

exploits). *Id.*, [0037] (“receive information regarding cyberthreats”), [0040]. “With this information, the synthetic test generator component 514 ... determines ... one or more steps of an instruction for emulating that portion of the objective.” *Id.*, [0040], Fig. 5.

117. A POSITA would have understood that the steps for emulating each exploit would return one or more vulnerabilities associated with the application of each exploit. For example, a POSITA would have known that an exploit having a complex attack path (*e.g.*, “complex attack scenario[.]”) would comprise multiple steps for penetrating the system. Vallone (EX1004), [0019]; *see also, id.* [0019], [0040] (generally teaching complex, multi-step exploits).

118. Typically, each exploit’s attack path would have included at least one vulnerable component for infiltrating the system. *See also* §V.D above (discussing how correlating exploits and vulnerabilities via attack paths was a well-known method); *see also*, Carvalho (EX1023), 4 (explaining that steps in an attack follow a path of vulnerabilities thought of as “prerequisites that must be satisfied for the attack step to be successful”); Patsos (EX1006), 295-296 (creating a “sample attack path (a_1)” and correlates exploits (e_j) with vulnerabilities (v_k)). Patsos (EX1006), 295-296, Figure 1. Moreover, Patsos defines the attack path as “the unique sequence of successful vulnerability exploits[.]” and notes that every exploit may correspond to one or more vulnerabilities, and vice versa. *Id.*, 295. “Therefore, the attack path

a_i is the series of v_k exploited by an attacker using e_j .” *Id.*

119. Indeed, Vallone describes an example exploit attack path, including instructions that require both execution of a malicious Flash script and the establishment of a data exfiltration tunnel. Vallone (EX1004), [0040]. This example involves exploitation of a client-side vulnerability (*e.g.*, a Flash Player exploit) and a separate network-level vulnerability enabling unauthorized outbound communication. *Id.* (describing multiple steps of an exploit, including “infiltrating a network via running a Flash browser-based script “badscript.fl” and creating a tunnel between data 520 and an Internet server”). Vallone’s system thus inputs an exploit and returns one or more vulnerabilities by constructing the steps for emulating the exploit, each set of steps typically including one or more steps for exploiting a vulnerability found in one or more vulnerable components of the computer system.

120. Vallone’s teachings are consistent with the ’385 Patent’s description of its “exploit function” which maps the attacker’s strategy. *See, e.g.*, ’385 Patent (EX1001), 3:61-63 (“The set of vulnerabilities produced by [exploit function] $ExF(A)$, for a given set of exploits A , represents the vulnerabilities that are exploited by the exploits in A ”), 4:9-11 (“identifies a set of vulnerabilities that he is willing to expose (often referred to as the ‘presented attack surface’”), 4:53-54 (“vulnerabilities generated by an attack strategy’s exploits”), 10:6-9 (“An exploit

function (denoted Ex) maps the attacker's strategy to a set of vulnerable software applications, which can be easily obtained from darknet marketplace data"); *see also id.*, 3:59-4:5 ("many possible variations of an exploit function are possible"). Similarly, Vallone maps an attack path of an exploit with instructions for emulating the attack in a synthetic test, where this attack path identifies vulnerable components and/or software exploited by the vulnerability.

121. **In sum**, Vallone teaches "applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities [Vallone's determining each exploit's series of steps in an attack path, and the vulnerabilities involved in that attack path]."

9. *Claim 1(g): "creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system;"*

122. I have been informed that Skysong accuses CrowdStrike's Falcon Platform and specifically the Falcon Sandbox module of infringing this limitation. Infringement Contentions, EX1030, 53-58. To "creat[e] a constraint set of the computer system from the set of vulnerabilities," I understand that Skysong alleges that Falcon Sandbox "establishes a space where potentially harmful code can be safely executed' ... in which 'investigators can ensure the rest of the system remains unaffected, thereby minimizing the risk of a security breach.' For example, '[a]pplication-level sandboxes restrict an application's access to certain system

resources. These provide enough resources for testing malicious code while still preventing that code from causing wider damage.” *Id.*, 53-54.

123. Skysong further contends that by “creat[ing] a controlled and isolated environment[,]” Falcon Sandbox’s alleged “constraint set compris[es] a minimum set of dependencies to operate the computer system.” *Id.* That is, I understand that Skysong accuses that constraining harmful code in a controlled, isolated environment is the claimed “creating a constraint set[.]” To the extent that the claims are interpreted and “creating a constraint set of the computer system from the set of vulnerabilities” such that “the constraint set compris[es] a minimum set of dependencies to operate the computer system” includes testing malicious code against computer vulnerabilities in an environment with enough resources to test the malicious code while preventing wider damage, Vallone teaches this limitation.

124. Vallone’s system “creat[es] a constraint set of the computer system from the set of vulnerabilities” by “synthesiz[ing] the cyberthreat (e.g., harvest.pl) as executable instructions that can be limited so that the ultimate objective of the cyberthreat ... is not committed[.]” Vallone (EX1004), [0038]; *see also, id.*, [0014] (teaching “methods, devices, and systems ... related to generating synthetic security events, representative of actual cyberthreat activity, for testing the effectiveness of a network’s detective and preventative information security controls”). To accomplish this, Vallone’s threat simulation technology’s server 510 instantiates

agents on host computers and instructs the agents to “executes steps of the instructions as if the agent 350 were the actual cyberthreat (e.g., a malware agent will mimic the characteristics/actions of real malware).” Vallone (EX1004), [0038].

125. In other words, by mimicking the sequence of events of an attack on the host computer against certain vulnerabilities, while preventing the ultimate malicious objective, Vallone is “creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system.” *See also* §V.F above (discussing how testing exploits in isolated, controlled environments was commonplace).

126. For example, Vallone “creat[es] a constraint set of the computer system from the set of vulnerabilities” by testing malicious code against computer vulnerabilities in an environment. Namely, Vallone synthetically tests exploits against vulnerabilities by “synthesiz[ing] the cyberthreat (e.g., harvest.pl) as executable instructions that can be limited so that the ultimate objective of the cyberthreat ... is not committed[.]” Vallone (EX1004), [0038].

127. Vallone’s “constraint set compris[es] a minimum set of dependencies to operate the computer system” because the “executable instructions [] can be limited so that the ultimate objective of the cyberthreat (e.g., posting stolen credit card numbers for sale on the Internet) is not committed[.]” Vallone, [0038]; *see also*, *id.*, [0040] (teaching emulating a cyberthreat “without the hazard of the objective

actually being achieved[.]” such as “steal[ing] usernames and passwords”). A POSITA would have understood that in this way, Vallone’s system is restricting access at the host for posting credit card information on the Internet in the same way that Skysong accuses testing malicious code while preventing wider damage. Infringement Contentions (EX1030), 53-54.

128. **In sum**, Vallone’s system “creat[es] a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system [Vallone tests malicious code against computer vulnerabilities in an environment with enough resources to test the malicious code while preventing wider damage]”.

10. Claim 1(h): “applying the set of exploits to the constraint set of the computer system;”

129. Once the instructions are compiled for simulating the exploit, the synthetic test is executed (*i.e.*, applying the set of exploits to the constraint set of the computer system). *See, e.g.*, Vallone (EX1004), [0041] (“a test is executed, at step 612”), Fig. 6. Vallone’s “synthetic test” “synthesize[s] the cyberthreat (e.g., harvest.pl) as executable instructions that can be limited so that the ultimate objective of the cyberthreat (e.g., posting stolen credit card numbers for sale on the Internet) is not committed, while still providing an administrator with knowledge (e.g., via feedback from agent(s) 350 and/or network equipment 340) regarding the depth of vulnerability of the target network 300 to an actual attack.” *Id.*, [0038].

130. For example, “synthetic test generator component 514, in various embodiments, determines, for at least a portion of a cyberthreat’s objective, one or more steps of an instruction for emulating that portion of the objective.” *Id.*, [0040]. “[I]f a cyberthreat’s objective is to steal usernames and passwords and includes steps for infiltrating a network via running a Flash browser-based script ‘badscript.fl’ and creating a tunnel between data 520 and an Internet server, then synthetic test generator 514 can determine corresponding instructions for execution by an agent 350 at a host 320 of a target network 300 for simulating that objective, without the hazard of the objective actually being achieved.” *Id.*

131. In this way, Vallone’s threat simulation technology “appl[ies] the set of exploits to the constraint set of the computer system” by installing and instructing agents to execute the instructions at the host computer. Specifically, threat simulation technology 500 “installs and/or instantiates **one or more agents** 350 on a host 320.” Vallone (EX1004), [0038]. These agents “are configurable by the technology to execute instructions on a host 320 for performing characteristics of a cyberthreat[.]” *Id.*; *see also, id.*, (agents “contain[] features that when configured with instructions can synthesize the characteristics of the cyberthreat”). This includes running “several simultaneous synthetic tests at a [] time” for multiple exploits. Vallone (EX1004), [0041]. A POSITA would have understood that running one set of instructions is consistent with running an exploit. Therefore,

running more than one set of instructions at a host computer simultaneously, as taught by Vallone, is equivalent to applying more than one exploit (*i.e.*, set of exploits) at a host computer at a time. Vallone (EX1004), [0038] (installing “one or more agents”), [0041] (running “simultaneous tests”).

132. In addition to running simultaneous synthetic tests, Vallone teaches running more than one set of instructions (*i.e.*, set of exploits) during a single synthetic test by modifying the synthetic test in real time to perform different objectives and therefore running different tests sequentially via a given agent. For instance, Vallone teaches that “[a] new and/or improved plan/strategy ... for implementing the cyberthreat’s objectives can ... be determined based on the feedback for automatically ... configuring or **reconfiguring agents** during and/or after each step of an instruction, based on a desired behavior (e.g., modify known cyberthreat objectives with additional potential objectives to determine whether adaptations of the cyberthreat can affect a target network[.]” Vallone (EX1004), [0019].

133. A POSITA would have understood that by actively reconfiguring and modifying a synthetic test in real-time to achieve additional objectives, Vallone executes different exploit code. Lastly, a POSITA would have understood that Vallone discloses the capability of executing both concurrently running agents (each testing at least one exploit), as well as reconfiguring any agent to execute additional

exploits sequentially.

134. Because Vallone’s threat simulation technology is responsible for both installation/instantiation of the agents, as well as providing the instructions for which vulnerabilities to exploit and when to exploit them, its processor performs this step. Said another way, Vallone’s agents are non-autonomous software entities that act only upon receiving explicit instructions from a coordinating system, which in this case is Vallone’s threat simulation technology. Given that, a POSITA would have understood that Vallone’s threat simulation technology performs the operation of “applying the set of exploits to the constraint set of the computer system[.]”

135. **In sum**, Vallone teaches “applying the set of exploits [Vallone running the synthetic test via agents on hosts] to the constraint set of the computer system [Vallone running the synthetic test in a protected environment, the synthetic test having executable instructions that can be limited so that the ultimate objective of the cyberthreat is not committed].”

11. Claim 1(i): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”

136. Vallone’s system includes “sensor component 416” which “monitors target network 300 activity during a synthetic test and passes feedback data back to ... threat simulation technology 500. Vallone (EX1004), [0039]. The “[f]eedback data (e.g., real-time and/or post-instruction data) from the agent(s) 350 and/or

network equipment 340 ... is **sent to and/or accessed by the threat simulation technology** [500], which can further **analyze the data[.]**” *Id.*, [0038]. To this end, Vallone’s threat simulation technology includes “validation feedback component 516” which “receives the feedback data ... from the target network 300. The feedback data can be stored as data 520 and used by the other components 512 (e.g., the synthetic test policy component 515) to improve the efficiency of **detecting**, assessing, responding to, and fixing **vulnerabilities**.” Vallone (EX1004), [0040]. Vallone specifies that “if an instruction has multiple steps ... each step can send feedback regarding its progress (e.g., whether the step was successful, failed, was interrupted, etc.).” *Id.*, [0019].

137. As such, Vallone’s threat simulation technology analyzes the application of the exploits by analyzing the feedback data and detects a particular vulnerability if execution of a certain step was successful.

138. **In sum**, Vallone teaches “analyzing the application of the set of exploits on the computer system [Vallone’s collected feedback data during the synthetic test] to detect a particular vulnerability of the computer system [Vallone using the feedback data to detect a particular vulnerability].”

12. Claim 1(j): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”

139. Vallone teaches that “[b]ased on the analysis of the validation feedback

data ... the technology can dynamically update the target network's security capabilities to, e.g., protect against the same and/or another vulnerability.” Vallone (EX1004), [0020]. Vallone calls these dynamic updates “self-healing process[es]” and provides examples such as “downloading and/or installing updates, firmware, new releases, antivirus rules, etc.” *Id.* Based on feedback data, “the target networks 300 can be (re)provisioned (e.g., **by updating servers 310 and hosts 320 with recent security updates**)[.]” *Id.*, [0040] (“[t]he feedback data can be stored as data 520 and used by the other components 512 (e.g., the synthetic test policy component 515) to improve the efficiency of detecting, assessing, responding to, and fixing vulnerabilities”). *See also* §V.G above (discussing how remediation of computer systems, e.g., by patching, was a commonplace technique used to fix vulnerabilities based on results of an exploitation test).

140. **In sum**, Vallone teaches “altering a configuration of the computer system [Vallone updating hosts] in response to the analysis of the application of the set of exploits [Vallone basing on the feedback data from the synthetic test] to reduce potential damage of a cyberattack [Vallone protects the computer system from future cyberattacks].”

13. Claim 8(Pre): “A method for improving a computer device, the method comprising:”

141. To the extent the preamble is limiting, Vallone teaches “[a] method for improving a computer device[.]” Vallone (EX1004), [0020]-[0021] (teaching its

process “dynamically update[s] the target network’s security capabilities to, e.g., protect against the same and/or another vulnerability”), Claims 1-23 (teaching “[a] method,” and more specifically “a computer-implemented method”); *see also* §VI.A.2 above, Claim 1(Pre). Protecting a computer device from vulnerabilities improves the computer device.

14. Claim 8(a): “accessing data comprising dark net information associated with a computer system;”

142. As an initial matter, a POSITA would have understood that there is no meaningful difference between Claim 8(a)’s “accessing” and Claim 1(d).

143. Vallone teaches “accessing data comprising dark net information[.]” Vallone (EX1004), [0005] (teaching accessing “the dark net [] to determine whether a new cyberthreat has been released and, if so, what its capabilities and objectives are”); *see* §VI.A.6 above, Claim 1(d). The cyberthreats (*i.e.*, dark net information) are “associated with a computer system[.]” Vallone (EX1004), [0015] (describing “[h]osts” as “laptops” that “can be affected by a cyberthreat”).

15. Claim 8(b): “obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of the computer system;”

144. As an initial matter, a POSITA would have understood that there is no meaningful difference between Claim 8(b)’s “bypass” and Claim 1(e) “penetrate”: both refer to exploits that intrude or break into computer systems and software.

145. Vallone teaches “obtaining a set of exploits from the dark net

information[.]” See §VI.A.7 above, Claim 1(e). These “set of exploits [are] configured to bypass a security feature of the computer system[.]” Vallone (EX1004), [0040] (describing exemplary scenarios in which the cyberthreat bypasses a security system, such as “steal[ing] usernames and passwords”); see §VI.A.7 above, Claim 1(e).

16. Claim 8(c): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”

146. Vallone applies an exploit function which takes the set of exploits as input and returns a set of vulnerabilities (*i.e.*, vulnerable components and/or software in an exploit attack path). See §VI.A.8 above, Claim 1(f).

17. Claim 8(d)(1): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system,”

147. Vallone teaches “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system” by testing malicious code against computer vulnerabilities in an environment with enough resources to test the malicious code while preventing wider damage. See §VI.A.9 above, Claim 1(g) (comparing Vallone to Skysong’s infringement allegations in litigation); *compare* Infringement Contentions (EX1030), 53-58 with 120-127 (alleging Falcon Sandbox infringes Claims 1(g) and 8(d)(1) in the same way).

18. Claim 8(d)(2): “wherein application of the set of exploits on the computer system comprises determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system;”

148. Vallone teaches applying the set of exploits. See §VI.A.10 above, Claim 1(h). Vallone further teaches “determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system[.]” See, e.g., Vallone (EX1004), [0038]-[0039] (teaching a sensor for monitoring the network activity during the synthetic test and passing feedback data to threat simulation technology for analysis).

19. Claim 8(e): “analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system; and”

149. Vallone teaches “analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system[.]” See §VI.A.11 above, Claim 1(i). “[I]f an instruction has multiple steps ... each step can send feedback regarding its progress (e.g., whether the step was successful, failed, was interrupted, etc.)” such that Vallone analyzes the feedback data and detects a particular vulnerability if execution of a certain step was successful. Vallone (EX1004), [0019]; see §VI.A.11 above, Claim 1(i).

20. Claim 8(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”

150. Vallone alters a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack by “updating ... hosts 320 with recent security updates” based on feedback data. Vallone (EX1004), [0040]; *see* §VI.A.12 above, Claim 1(j).

21. Claim 14: “The method of claim 8, wherein altering the configuration of the computer system comprises installing a patch to a software component of the computer system that is exposed by the set of exploits to attack the computer system.”

151. Vallone teaches altering the configuration of the computer system. Vallone (EX1004), [0040] (teaching, based on feedback, “updating ... hosts 320 with recent security updates”); *see* §VI.A.20 above, Claim 8(f), and §VI.A.12 above, Claim 1(j). This includes “downloading and/or installing updates, firmware, new releases, antivirus rules, etc. ... [which] alleviates the time-consuming measures ... when there are hundreds of computing devices[.]” Vallone (EX1004), [0020]; *see also, id.*, [0022] (teaching healing the network includes “addressing the vulnerability”).

152. **In sum**, Vallone teaches “installing a patch [Vallone’s updates, firmware, new releases] to a software component of the computer system that is exposed by the set of exploits to attack the computer system.”

22. Claim 15(Pre): “One or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine, the computer process comprising:”

153. To the extent the preamble is limiting, Vallone teaches “[o]ne or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine[.]” Vallone (EX1004), [0028] (teaching “computer-executable instructions, such as routines executed by a ... processing device” “stored ... on tangible computer-readable media”), Claims 10-23 (teaching “[a] computer-readable storage device having instructions that, when executed by at least one processor, causes the at least one processor to perform a computer-implemented method”); *see* §VI.A.5 above, Claim 1(c).

23. Claim 15(a): “obtaining a set of exploits from dark net information, the set of exploits configured to bypass a security feature of a computer system;”

154. Vallone teaches obtaining exploits from dark net information which are configured to bypass a security feature. *See* §VI.A.7 above, Claim 1(e), and §VI.A.15 above, Claim 8(b) (identical language).

24. Claim 15(b): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”

155. Vallone teaches applying an exploit function as claimed. *See* §VI.A.8 above, Claim 1(f) and §VI.A.16 above, Claim 8(c) (identical language).

25. Claim 15(c): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system;”

156. Vallone teaches creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system. See §VI.A.17 above, Claim 8(d)(1) (identical language), and §VI.A.9 above, Claim 1(g).

26. Claim 15(d): “applying the set of exploits to the constraint set of vulnerabilities of the computer system;”

157. See §VI.A.10 above, Claim 1(h).

27. Claim 15(e): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”

158. Vallone teaches analyzing the application of the set of exploits as claimed. See §VI.A.11 above, Claim 1(i), and §VI.A.19 above, Claim 8(e).

28. Claim 15(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”

159. Vallone teaches altering a configuration of the computer system as claimed. See §VI.A.12 above, Claim 1(j), and §VI.A.20 above, Claim 8(f) (identical language).

VII. OPINION – GROUND 2

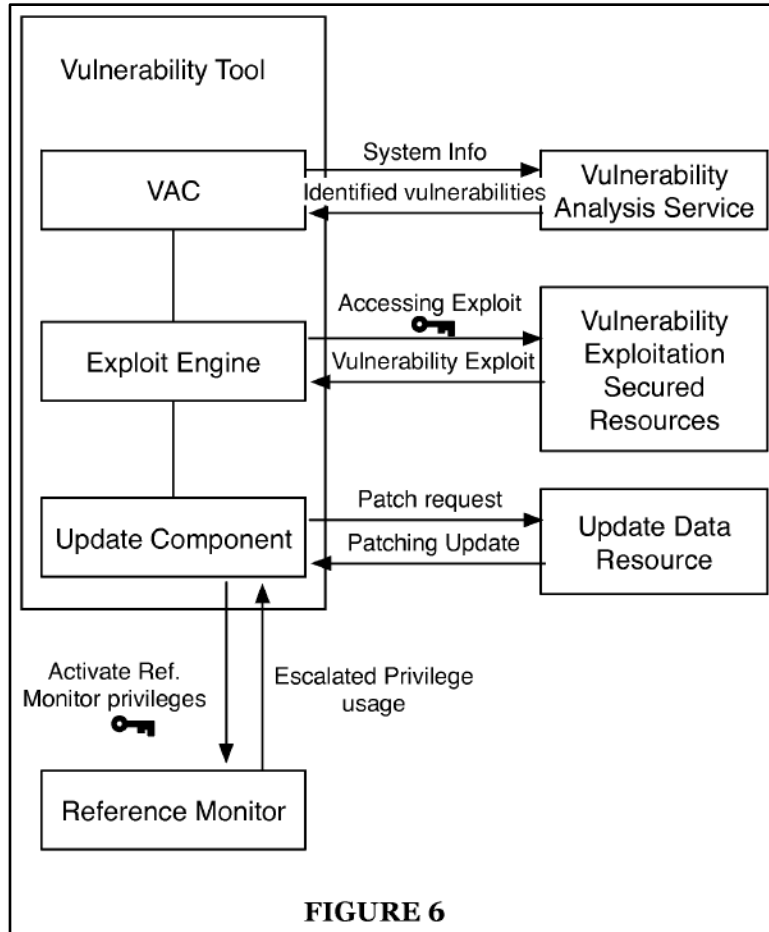
160. Claims 1, 8, 14, and 15 would have been obvious over Oberheide (EX1005), Patsos (EX1006), and Roytman (EX1007).

A. Obvious to Combine Oberheide, Patsos, and Roytman

1. Overview of Oberheide

161. U.S. Patent Publication No. 2014/0245450 to Oberheide *et al.* (“Oberheide”) was filed February 24, 2014, and published on August 28, 2014 (EX1005).

162. Oberheide teaches a “system and method that includes identifying a vulnerability in a computing device; accessing a vulnerability exploitation mapped to the identified vulnerability; at the computing device, executing the vulnerability exploitation and ... updating the system with a vulnerability resolution.” Oberheide (EX1005), Abstract, [0013], [0019]. This process is depicted below in Figure 6:



Id., Fig. 6.

163. Oberheide teaches a “vulnerability tool” downloadable to a device for performing the described functions. *Id.*, [0014], Fig. 6. Oberheide’s vulnerability tool includes a “vulnerability assessment component (VAC)” that identifies vulnerabilities using “any suitable approach,” including “malware detection approaches.” *Id.*, [0015], [0020]. The vulnerability tool also includes an exploit engine that accesses “exploitation scripts” to exploit vulnerabilities. *Id.*, [0017]. An update component then applies “a patch or system update to resolve or reduce an existing vulnerability.” *Id.*, [0018], [0024], [0026].

164. Oberheide is in the same field of endeavor as the '385 Patent because both are directed to systems and methods for exploiting patching vulnerabilities. *Compare* Oberheide (EX1005), Abstract (teaching a system and method comprising “executing the vulnerability exploitation and [...] updating the system with a vulnerability resolution.”), [0013] (“a system for patching a device through exploitation” that may include “an exploit engine, and an update component.”), [0017] (“The exploit engine of a preferred embodiment functions to execute an exploitation process[,]” wherein “[t]he exploit engine preferably uses at least one of the vulnerabilities identified”), [0019], [0026], Figs. 5-8 *with* '385 Patent (EX1001), 3:4-15 (describing “penetration testing that is fed with real-world exploit market data”), 3:61-64 (describing exploiting a set of vulnerabilities), 13:12-18 (reverse engineering exploits “to develop a patch to the exposed vulnerability), Figs. 5 and 7.

165. The '385 Patent describes common problems in the art related to developing patches for system vulnerabilities. '385 Patent (EX1001), 13:55-56, 14:37, 15:17; *see also id.*, 2:63-3:3 (describing the defender of a target computer system wanting to “minimize the effectiveness of an attacker”), 13:15-18. Oberheide similarly addresses the “need in the mobile security field to create a new and useful system and method for patching a device through exploitation.” Oberheide (EX1005), [0003]; *see also id.*, Abstract, [0013] (“The system is preferably used to enable capabilities to fix, update, and/or reduce the vulnerabilities

of a system.”), [0019]. As such, Oberheide is reasonably pertinent to the problems addressed by the ’385 Patent and is analogous art.

2. *Overview of Patsos*

166. Patsos published in Information Management & Computer Security in October 2010. Patsos (EX1006).

167. Patsos teaches an “incident response intelligence system (IRIS)” that is a security-management platform that “models the context of discovered vulnerabilities, calculates their significance, finds and analyzes potential exploit code and defines the necessary intrusion detection signatures that combat possible attacks, using standardized techniques.” Patsos (EX1006), 291, 293. To accomplish this, IRIS “collects security information related to vulnerabilities,” including real world exploit information from various sources including “underground web sources” and “hacking web sites[.]” *Id.*, 295-297 (“[i]nformation related to exploits (and actual exploit code) can be obtained from hacking web sites” using an “exploit gathering module”). IRIS then uses a “vulnerability exploit correlation module (VECM)” to correlate exploits to vulnerabilities and countermeasures. *Id.*, 298 (describing “providing a full correlation between all attack components, i.e.,[.] the vulnerabilities, the exploits that can potentially realize these vulnerabilities and the associated countermeasures”).

168. A POSITA would have understood that Patsos’s “underground web

sources” and “hacking web sites” are dark net information sources because they contain the same (*e.g.*, illicit) information.

169. Patsos is in the same field of endeavor as the '385 Patent because both are directed to systems and methods for exploiting vulnerabilities and patching vulnerabilities. *Compare* '385 Patent (EX1001), 3:4-15, 3:61-64, 13:12-18 (reverse engineering exploits “to develop a patch to the exposed vulnerability[,]” Figs. 5 and 7 *with* Patsos (EX1006), 292, 295, 300 (describing a “main objective of this process is to enhance the TVA [topological vulnerability analysis] by providing a complete mapping between vulnerabilities, exploits and signatures that should describe not only the attack path an attacker can use (by exploiting a series of vulnerabilities that are present in different system boundaries, i.e. in the perimeter and in the internal systems) but also a response path that should describe the corresponding response policies that have to be enforced in order to detect – and perhaps block – these actions.”).

170. The '385 Patent describes common problems in the art related to improving the security of a computer system or device. '385 Patent (EX1001), 2:63-3:3 (describing the defender of a target computer system wanting to “minimize the effectiveness of an attacker”), 13:15-18. Patsos similarly addresses “identify[ing] the infrastructure or related components that may be at risk from attacks and the determination of appropriate procedures that can be implemented to reduce this level

of risk[.]” Patsos (EX1006), 292, 293 (introducing the “incident response intelligence system” for vulnerability analysis and intrusion detection), 296, 303. As such, Patsos is reasonably pertinent to the problems addressed by the ’385 Patent and is analogous art.

3. Overview of Roytman

171. U.S. Patent Publication No. 2015/0237065 to Roytman *et al.* (“Roytman”) was filed on March 9, 2015, and published on August 20, 2015 (EX1007).

172. Roytman teaches techniques for “ranking a set of vulnerabilities of a computing asset and set of remediations for a computing asset[.]” Roytman (EX1007), Abstract. “In one technique, vulnerabilities ... are received at a vulnerability intelligence platform” along with “[b]reach data indicating set of breaches that occurred[.]” *Id.* Then, a “subset of the set of vulnerabilities that are most vulnerable to a breach is identified[.]” a “risk score is generated[.]” and “remediations are ordered based on the remediations that would reduce the risk score the most[.]” *Id.*; *see also id.*, [0025], Fig. 3.

173. Specifically, in Roytman (EX1007), vulnerabilities are identified from one or more computing assets in a customer network. *Id.*, [0025], Fig. 3. The computing asset may be one that is “absolutely critical” for day-to-day operations. *Id.*, [0044]. A server then receives data regarding breaches (*e.g.*, successful exploits)

to determine “[a] subset of the set of vulnerabilities that are most vulnerable to the breach[.]” *Id.*, [0025], [0032]. A risk score is assigned to the vulnerabilities. *Id.*, [0026]-[0027], [0034]. Using the vulnerabilities and their corresponding risk scores, remediations are selected to “reduc[e] the risk posed by one or more vulnerabilities of a computing asset or a group of computing assets.” *Id.*, [0092]-[0093], [0100].

174. Roytman is in the same field of endeavor as the ’385 Patent because both are directed to systems and methods for exploiting vulnerabilities and patching vulnerabilities. *Compare* Roytman (EX1007), [0027], [0030] (using breach data to determine “the likelihood that an exploit of a particular vulnerability of a particular asset will be successful”), [0032] (“a breach is a successful attack on a computing asset by successfully exploiting a vulnerability of the computing asset”), [0038], [0060]-[0061], [0069], [0092] (describing selecting remediations for the exploited vulnerabilities) *with* ’385 Patent (EX1001), 3:4-15 (describing “penetration testing that is fed with real-world exploit market data”), 3:61-64 (describing exploiting a set of vulnerabilities), 13:12-18 (reverse engineering exploits “to develop a patch to the exposed vulnerability), Figs. 5 and 7.

175. The ’385 Patent describes common problems in the art related to improving the security of a computer system or device. ’385 Patent (EX1001), 2:63-3:3 (describing the defender of a target computer system wanting to “minimize the effectiveness of an attacker”), 13:15-18. Roytman similarly addresses the problem

of improving the security of a computer system or device. Roytman (EX1007), [0002], [0025]-[0027], [0038], [0100] (“The embodiments described herein may enable for an assessment of risk associated with a computing asset and provide an effective and efficient manner in reducing the risk posed by one or more vulnerabilities of a computing asset or a group of computing assets.”). As such, Roytman is reasonably pertinent to the problems addressed by the ’385 Patent and is analogous art.

4. *Claim 1(Pre): A system for protecting a computer system from attack, the system comprising:*

176. As an initial matter, I note that claim 1 is not limited to the claimed “computer system” itself as executing all the claimed functions below.

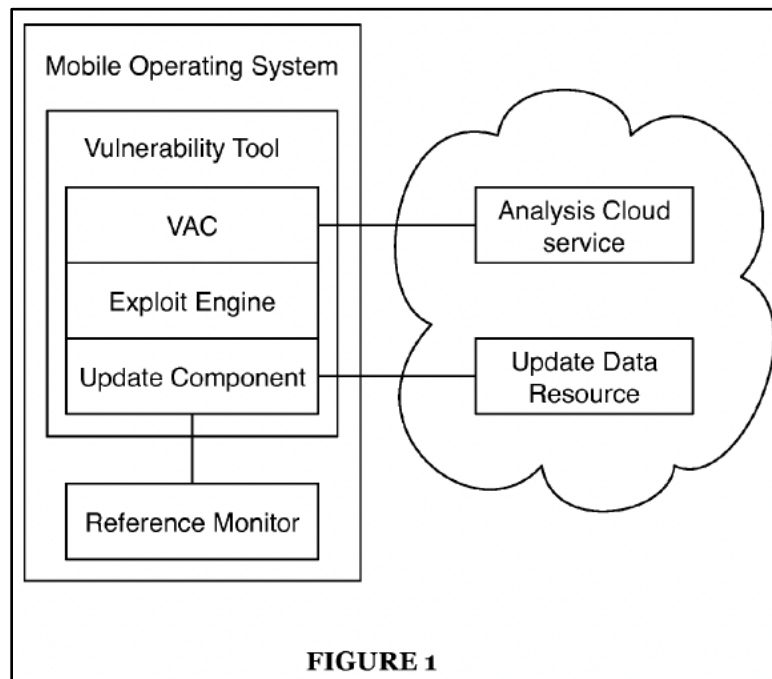
177. To the extent the preamble is limiting, Oberheide discloses a system for protecting a computer system from attack. Oberheide (EX1005), [0013] (disclosing “a **system** for patching a device”). Oberheide’s system is used to “reduce the vulnerabilities of a system” and “apply a persistent fix on a **computer system**[.]” *Id.* A vulnerability is “a flaw or weakness” of a system that risks exploitation in an attack. Oberheide (EX1005), [0003]. By reducing weaknesses and applying persistent fixes to a computer system, Oberheide’s teaches “[a] system for protecting a computer system from attack[.]”

178. **In sum**, Oberheide teaches “[a] system for protecting a computer system from attack [Oberheide reducing vulnerabilities on a system and applying a

persistent fix].”

5. Claim 1(a): a network connection;

179. Oberheide teaches its system is connected to a network. Oberheide (EX1005), [0016], Fig. 1 below:



Id., Fig. 1.

180. For example, an “analysis cloud service may communicate with the VAC using any variety of protocols using any type of network interface deemed appropriate.” Oberheide (EX1005), [0016] (“Such a network may be public or private in nature”).

181. **In sum**, Oberheide teaches “a network connection [Oberheide’s network interface].”

6. Claim 1(b): a processing device; and

182. Oberheide teaches “a general or application specific processor” that executes instructions for performing Oberheide’s embodiments. Oberheide (EX1005), [0031] (specifying that the instructions are “executed by computer-executable components preferably integrated with the vulnerability tool” of the computing device, such that the “computer-executable component is preferably a general or application specific processor.”).

183. **In sum**, Oberheide teaches “a processing device [Oberheide’s general or application specific processor executing instructions and integrated with the vulnerability tool].”

7. ***Claim 1(c): a non-transitory computer-readable medium connected to the processing device configured to store instructions that, when executed by the processing device, performs the operations of:***

184. Oberheide teaches its “system and methods of the preferred embodiment and variations thereof can be embodied and/or implemented at least in part as a machine configured to receive a computer-readable medium storing computer-readable instructions.” Oberheide (EX1005), [0031]. “The instructions are preferably executed by computer-executable components” and the “computer-executable component is preferably a general or application specific processor[.]”

Id. Oberheide teaches that “[t]he computer-readable medium can be stored on any suitable computer-readable media such as RAMs, ROMs, flash memory, EEPROMs, optical devices (CD or DVD), hard drives, floppy drives, or any suitable

device.” *Id.* At least some of these media are non-transitory (e.g., ROMs, flash memory, EEPROMs, optical devices, hard drives, and floppy drives).

185. **In sum**, Oberheide teaches “a non-transitory computer-readable medium [Oberheide’s computer-readable medium such as ROM] connected to the processing device [Oberheide’s general or application specific processor] configured to store instructions [Oberheide’s computer-readable instructions] that, when executed by the processing device, performs the [claimed] operations[.]”

8. *Claim 1(d): accessing data through the network connection, the data comprising dark net information associated with exploits of a computer system, comprising:*

186. As an initial matter, a POSITA would have recognized that there is nothing special about this limitation’s use of “dark net” information because the claim would operate the same on public, non-dark net data. *See* §V.B above.

187. Oberheide teaches its system is connected to a network. *See* §VII.A.5 above, Claim 1(a); Oberheide (EX1005), [0016], Fig. 1. Oberheide also teaches a process for “access[ing] a resource storing a collection of exploitation scripts or routines.” Oberheide (EX1005), [0017].

188. While Oberheide does not expressly state that it accesses dark net exploit data over a network connection, Patsos teaches this.

189. Patsos teaches a security management system that “collects security information related to ... exploits ... from a large number of sources” for local

storage, including “underground web sources” such as “hacking web sites[.]” Patsos (EX1006), 295 (“Information regarding exploits can be found [] directly in a large number of underground web sources”), 296 (“hacking web sites”).

190. A POSITA would have understood that hacking web sites on underground web sources include “dark net information associated with exploits[.]” Namely, the dark net is an “underground web source[.]” and hacking web sites on the dark net include information associated with exploits of a computer system. *See, e.g.,* Chertoff (EX1022), 5 (citing “Exploit Markets” among types of cybercrime available to hackers on the dark web); Europol (EX1032), 17 (describing the growth of “cybercrime threats such as underground forums and malware” on the “Darknets”); *see also* §V.B above (discussing how it was common to obtain exploits from the dark net for use in penetration tests).

a. Motivation to Combine Oberheide with Patsos (Dark Net Data)

191. It would have been obvious, and a POSITA would have been motivated, to modify Oberheide to update its local repository of exploit scripts with exploits from the dark net as taught by Patsos for at least two reasons.

192. **First**, adding dark net information associated with exploits of a computer system to Oberheide’s local repository of exploits would enhance Oberheide’s security coverage by providing it with more useful information to process. Oberheide’s system would beneficially access the latest exploits being

developed and marketed by malicious actors on the dark net in order to patch vulnerabilities associated with those exploits. Chertoff (EX1022), 5 (explaining that “[e]xploit markets serve as platforms for buying and selling zero-day exploits,” “which no official patch has been released by the vendor. ‘Zero-day’ refers to the fact that the programmer has had zero days to fix the flaw.”); *see also*, Murthy (EX1033), [0018] (describing “0-day attacks that exploit previously unknown vulnerabilities.”). A POSITA would have understood that patching unknown vulnerabilities (*e.g.*, using zero-day exploits) would have improved Oberheide’s system by proactively getting rid of vulnerabilities before a new exploit is released and begins attacking and compromising systems.

193. **Second**, enhancing access to a broader selection of exploits via the dark net would have provided additional tools (*i.e.*, exploits) to gain the necessary privileges for patching vulnerabilities. For example, a POSITA would have recognized that the dark net contains exploits that Oberheide’s system might otherwise not be able to easily access. In this case, finding more exploits would broaden Oberheide’s capabilities for gaining privileges to patch a vulnerability “in situations where necessary privileges to update the system are not present or do not facilitate customized and/or accelerated patching of a device.” Oberheide (EX1005), [0013].

194. Moreover, Oberheide recognizes many “different approaches” are

necessary to patch certain vulnerabilities, and that different “exemplary exploits” are used. *Id.*, [0025] (providing list of exemplary exploits, such as “leveraging a buffer overflow vulnerability to perform arbitrary privileged code execution,” and “us[ing] application-level privilege escalation, which can include confused deputy exploits and collusion exploits.”). Updating Oberheide’s exploit repository with dark net exploits expands the range of approaches Oberheide’s system may choose from, allowing it to resolve more vulnerabilities.

195. Indeed, a POSITA would have understood that real-world attacks were regularly used to determine if a system had a vulnerability and/or how well it would tolerate the attack. For example, the NIST recommendations discuss penetration testing, in which “real-world attacks [are mimicked] to identify methods for circumventing the security features of an application, system, or network. It often involves launching real attacks on real systems and data that use tools and techniques commonly used by attackers.” Scarfone (EX1011), 36. Furthermore, “[m]ost penetration tests involve looking for combinations of vulnerabilities on one or more systems[.]” *Id.*

196. As such, a POSITA would have understood that updating Oberheide’s exploit repository with real-world exploits from the dark net would have enhanced its ability to identify vulnerabilities, determine how well they are tolerated, and subsequently fix the vulnerabilities that were exploited for privileged access. A

POSITA would have additionally understood, based on the standardized method of using real-world exploits, that using real-world exploits (*e.g.*, those obtained on the dark net) is preferable to exploits generated elsewhere. The real-world exploits found on the dark net are more likely to be implemented than those produced by other means and not being distributed/sold on the dark net.

197. A POSITA would have appreciated a reasonable expectation of success making this modification. Oberheide already teaches a network connection (*see* §VII.A.5 above, Claim 1(a)) through which Oberheide's modified system would interface with the dark net, as taught by Patsos (EX1006), to obtain exploits. *See also*, Chertoff (EX1022), 2 (describing the dark net as a portion of the World Wide Web, which itself is accessed over the medium of the Internet), 3 (discussing how the "Deep Web[,] of which the "Dark Web" is a part of, is accessed by "[c]ustom Web crawlers," and also "[a] relatively known source for content that resides on the dark Web is found in the Tor network"). A POSITA would have understood that Oberheide's access to the internet would have provided it access to the dark net (*e.g.*, via the Tor network) through which it could use known web crawlers, such as in Patsos, to obtain data.

198. Indeed, to obtain exploits, Patsos utilizes dark net crawlers to locally store exploit data. For example, in describing IRIS, Patsos states that its "system collects security information related to vulnerabilities, exploits and IDP [intrusion

detection and prevention] signatures from a large number of sources (web, mailing lists, hacking web sites ...).” Patsos (EX1006), 295-296; *see also, id.*, 296-297 (describing how IRIS collects security information, including via its “vulnerability gathering module” and “exploit-gathering module”). A POSITA would have understood that the process of “gathering” information is no different than “crawling” to collect such data.

199. Patsos specifies that its gathering modules in total collect information from at least 45 different sources (18 vendor web sites, 12 coordination centers, 12 independent lists, two IDP systems, and “three different sources”) and maintain more than 51,000 pieces of information (33,000 vulnerabilities, 16,000 IDP signatures, and 2,000 exploits). *Id.*, 297.

200. A POSITA would have understood that such massive amounts of data collection, sorting, and maintenance would have required an automated system. And for collection in particular, a POSITA would have understood that such an automated collection process would be performed by an Internet crawler as these crawlers were a common method employed for online data collection. *See, e.g.*, Fu (EX1018), 1218 (describing implementation of “a focused crawler for Dark Web forums” that “stores and analyzes the forum collection”); *see also, e.g.*, Zhang (EX1034), 1-2 (describing incremental forum spidering of dark web forums, the “spiders” being “software programs that traverse the World Wide Web information

space by following hypertext links and retrieving web documents by standard HTTP protocol” for “provid[ing] comprehensive Dark Web forum data from various sources in a timely manner.”); *see also* §V.A above (discussing the common use of web crawling for obtaining data). As crawling the dark web was a common technique for collecting large datasets, a POSITA would have understood that Patsos’s system similarly uses well-known crawlers to crawl the dark net (*e.g.*, hacking web sites) to access and obtain exploit data.

201. Given that dark web crawling was a well-known technique utilized in Patsos’s system, it would have been a simple addition well within the skill of a POSITA to implement Patsos’s crawler in Oberheide’s system to update Oberheide’s local exploit repository.

202. Furthermore, a POSITA would have understood that processing the dark net data for exploit information is substantially similar to processing data from the public web. Indeed, dark net data is not fundamentally different than any other exploit data, regardless of whether it came from the dark net or not. Any system that can process non-dark net data (*e.g.*, Oberheide) can be trivially adapted to process dark net data. Because of this, a POSITA would have had a reasonable expectation of success in modifying Oberheide to access and obtain the exploit data from the dark web, as taught by Patsos.

203. Taken together, Oberheide’s system includes a network connection for accessing information from online sources, including the analysis cloud service (*i.e.*, “accessing data through the network connection”). As modified by Patsos (EX1006), Oberheide’s system advantageously collects information related to exploits from underground web sources, such as hacking web sites (*i.e.*, “the data comprising dark net information associated with exploits of a computer system”).

204. **In sum**, the Oberheide-Patsos combination teaches “accessing data through the network connection [Patsos collecting security information over Oberheide’s network interface], the data comprising dark net information associated with exploits of a computer system [Patsos’s gathering information regarding exploits from underground web sources and hacking web sites].”

9. *Claim 1(e): obtaining a set of exploits from the dark net information, the set of exploits configured to penetrate the computer system;*

205. As an initial matter, a POSITA would have recognized that there is nothing special about this limitation’s use of “dark net” information because the claim would operate the same on public, non-dark net data. *See* §V.B above.

206. The Oberheide-Patsos combination teaches obtaining a set of exploits from the dark net information. *See* §VII.A.8 above, Claim 1(d). Namely, Oberheide’s modified system utilizes a dark net crawler to obtain exploits to update its local exploit repository. *Id.*; Oberheide (EX1005), [0017], Fig 6; Patsos

(EX1006), 295-296 (“[i]nformation related to exploits (and **actual exploit code**) can be **obtained from hacking web sites**”), 299 (providing “direct link to the actual exploit”); *see also* §V.B above (discussing that obtaining data from dark net markets was commonplace).

207. These exploits are configured to penetrate the computer system. Oberheide’s system “identif[ies] vulnerabilities and exploit[s] such vulnerabilities to enable capabilities to fix the vulnerabilities.” Oberheide (EX1005), [0013]. To accomplish this, Oberheide uses the exploits to “[e]nter[] an operating mode of escalated privileges[,]” which will “grant[] the vulnerability tool the necessary privileges to patch the vulnerability.” *Id.*, [0024]-[0025]. In other words, Oberheide uses its stored exploits to penetrate its computer system to force a patch.

208. **In sum**, the Oberheide-Patsos combination teaches “obtaining a set of exploits from the dark net information [Patsos obtaining actual exploit code from hacking web sites], the set of exploits configured to penetrate the computer system [Oberheide using the exploits to enter an operating mode of escalated privileges].”

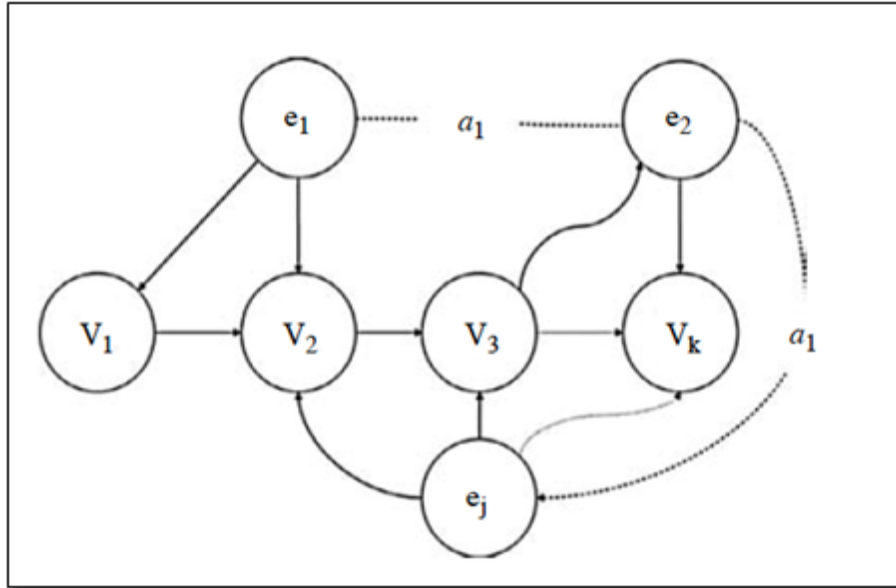
10. Claim 1(f): applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;

209. As an initial matter, a POSITA would have recognized that the term “function” in the ’385 Patent is not limited to a mathematical function such that it formulaically takes an input and returns an output, *e.g.*, $y = f(x_1, x_2, \dots)$. For example, a POSITA would have understood, based on the intrinsic record, that there

are “many possible variations of an exploit function [that] are possible.” ’385 Patent (EX1001), 3:64-4:5.

210. Oberheide teaches a vulnerability assessment component (VAC) that returns a set of vulnerabilities “us[ing] any suitable approach to identifying a vulnerability[.]” such as “anti-virus and/or malware detection approaches as would be known by those skilled in the art.” Oberheide (EX1005), [0015]; *see also id.*, [0019]-[0021], Fig. 6. While Oberheide teaches returning sets of vulnerabilities using known malware approaches (*e.g.*, exploitation of known weaknesses), Oberheide does not expressly discuss using an exploit function to return a set of vulnerabilities.

211. Patsos teaches a “vulnerability exploit correlation module (VECM)” that correlates exploits to vulnerabilities. Patsos (EX1006), 295-298 (“the correlation of exploits to vulnerability identifiers is performed by the vulnerability exploit correlation module VECM”). The VECM correlation module is “part[] of the system’s correlation engine.” *Id.*, 298. “This engine ... provid[es] a full correlation between all attack components, i.e. the vulnerabilities, the exploits that can potentially realize these vulnerabilities and the associated countermeasures[.]” *Id.* (describing taking an exploit and correlating the exploit to vulnerabilities using vulnerability identifiers). An example of this process is shown below in Figure 1:



Patsos (EX1006), 295.

212. Patsos’s system creates a “sample attack path (a_1)” and correlates exploits (e_j) with vulnerabilities (v_k). *Id.*, 295-296. In other words, Patsos’s VECM takes exploits as input and returns a set of vulnerabilities that correlate to those exploits. *See also* §V.C above (discussing how linking vulnerabilities to mined exploits was commonplace), §V.D above (discussing how correlating exploits and vulnerabilities via attack paths was a well-known method).

213. Patsos’s teachings are consistent with the ’385 Patent’s description of its “exploit function[.]” *See, e.g.*, ’385 Patent (EX1001), 10:6-9 (“An exploit function (denoted Ex) maps the attacker’s strategy to a set of vulnerable software applications, which can be easily obtained from darknet marketplace data”); *see also id.*, 3:59-4:5 (“many possible variations of an exploit function are possible”

including associating vulnerabilities with exploits listed in online marketplaces). Similarly, Patsos maps an attack path and correlates exploits with vulnerabilities.

214. **In sum**, the Oberheide-Patsos combination teaches “applying an exploit function [Oberheide’s VAC modified by Patsos’s VECM] which takes the set of exploits as input and returns a set of vulnerabilities [Patsos correlates vulnerabilities to exploits by creating a sample attack path].”

a. Motivation to Combine Oberheide with Patsos (Exploit Function)

215. It would have been obvious, and a POSITA would have been motivated to modify Oberheide’s VAC to implement Patsos’s VECM exploit function to identify the vulnerabilities associated with newly acquired exploits for patching for at least two reasons.

216. **First**, Oberheide teaches its “VAC may use any suitable approach to identify[] a vulnerability,” indicating a flexible, versatile architecture that can be extended easily. Oberheide (EX1005), [0015]. Patsos’s VECM correlation engine employs an exemplary “suitable approach to identifying a vulnerability” that constructs attack paths and correlates vulnerabilities to exploits. More specifically, Oberheide teaches several vulnerability assessment techniques, including leveraging a remote analysis cloud service or using a “local analysis engine” to identify vulnerabilities locally. Oberheide (EX1005), [0016].

217. A POSITA would have understood that Patsos's VECM exploit function would facilitate this local analysis using known and reliable correlation techniques. For example, a POSITA would have recognized the benefits of utilizing Patsos's computer-based correlation system over the prior methods. Indeed, Patsos discusses how tasks such as "correlate[ing] vulnerabilities with exploit code, or provid[ing] instruction detection and prevention (IDP) signatures" were at one time "human-oriented task[s], usually performed by high-skilled security analysts." Patsos (EX1006), 292. As such, a POSITA would have recognized the benefits of performing Patsos's VECM correlation in Oberheide, particularly when correlating exploits with vulnerabilities on its local analysis engine, as this would automate the process making it much faster and less error-prone than more archaic methods (*e.g.*, human-based correlating).

218. **Second**, a POSITA would have been motivated to modify Oberheide with Patsos to enhance patching vulnerabilities. Oberheide aims to provide a variation of exploits to gain device privileges to patch vulnerabilities. *See generally*, Oberheide (EX1005), [0025]. For example, Oberheide provides multiple different scenarios in which privileges differ. *Id.*, (teaching various privileges, such as "those for a superuser, root account, system administrator, kernel level account, higher privileged services/drivers, privileges of another user/application/service, and/or other suitable modes of increased permissions within the device").

219. A POSITA would have understood that correlating vulnerabilities to exploits using Patsos's VECM exploit function would improve Oberheide's process for gaining the different types of privileges needed for patching. For example, a POSITA would have recognized that correlating vulnerabilities with the exploits as taught by Patsos would have allowed Oberheide's system to more readily identify the vulnerabilities, and therefore gained privileges, accorded to a certain exploit. Therefore, the system would have a vulnerability-exploit correlation database to pull from when attempting to fix a vulnerability, the database including the privileges provided by the exploit. Knowing the privileges provided by an exploit would allow Oberheide's system to identify and apply exploits more efficiently based on the necessary privileges needed to fix an identified vulnerability.

220. A POSITA would have seen success when making this modification because Oberheide already teaches using "any suitable approach" to identify vulnerabilities and Patsos's VECM exploit function is a known suitable approach. As Oberheide's system was ready for modification to use any suitable approach, modifying it to implement Patsos's VECM exploit function would have required minimal software modifications within the skill of a POSITA.

221. **In sum**, Oberheide's VAC provides a component for identifying a vulnerability using any suitable approach, which, as modified by Patsos, uses a VECM exploit function that correlates vulnerabilities to exploits (*i.e.*, "appl[ies] an

exploit function which takes the set of exploits as input and returns a set of vulnerabilities”).

11. Claim 1(g): creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system

222. The Oberheide-Patsos combination teaches identifying sets of vulnerabilities in the computer system. *See* §VII.A.10 above, Claim 1(f); Oberheide (EX1005), [0015], [0019]-[0021], Fig. 6. As detailed in my opinions in this section, Oberheide creates a “constraint set” by constraining its vulnerability analysis to specific vulnerabilities based on: (1) vulnerabilities associated with a specific component of the system; or (2) system capabilities after exploitation of a vulnerability. While Oberheide does not expressly discuss how it constrains its analysis to these specific vulnerabilities, Roytman teaches a ranking technique to identify a subset of vulnerabilities most vulnerable (*i.e.*, amenable) to exploitation.

223. Oberheide teaches that “[p]referably, vulnerabilities are identified for an entire system, but the vulnerabilities may alternatively be a set of vulnerabilities for a particular component or set of components of a device such as for a service or an application within an operating system of a computing device.” Oberheide (EX1005), [0020]. For example, the constrained set of vulnerabilities may be a particular application in the system, such as “a browser, communication framework, or any suitable component[.]” *Id.*, [0020].

224. Moreover, Oberheide teaches constraining to specific vulnerabilities “based on the ... enabled capabilities after the exploitation[.]” *Id.*, [0024]. In other words, Oberheide constrains its vulnerability analysis to (1) vulnerabilities associated with a particular computer component, or (2) vulnerabilities that significantly alter the computer’s capabilities after exploitation. Oberheide therefore teaches or suggests creating a constraint set comprising “a minimum set of dependencies to operate the computer system” such that its vulnerability analysis may be constrained to vulnerabilities that shut down the computer system through successful exploitation and/or to vulnerabilities that render facets of the computer system inoperable. I note that the ’385 Patent’s specification does not explain how a constraint set comprises a “minimum set of dependencies to operate the computer system[.]” My interpretation is therefore consistent with the plain meaning of that terminology in light of the sparse intrinsic record.

225. Oberheide provides multiple examples of vulnerabilities that, if exploited, cause a user to lose control of certain aspects and even the entirety of the computer. For example, Oberheide teaches identifying vulnerabilities “that can be exploited for escalated privileges” such as “allowing a device to be rooted (e.g., ASHMEM vulnerability), neglecting to check if messages are from a trusted source (e.g., Exploids or Gingerbreak vulnerability), privilege levitating (e.g., Levitator vulnerability), writing to arbitrary memory address space (e.g., MempoDroid

vulnerability), overwhelming a system with command arguments (e.g., Android ZergRush vulnerability) neglecting to check return values enabling root shells (e.g., Zimmerlich vulnerability), and/or any suitable vulnerability or exploits.” Oberheide (EX1005), [0022].

226. Each of these listed vulnerabilities cause a user to fully or partially lose the ability to operate aspects of a computer. For instance, the ASHMEM exploit allows escalation of privileges to root (full administrator or super-user), providing the attacker with administrative/super-user access and therefore overriding normal user access controls. As another example, overwhelming a system with command arguments (such as with the Android ZergRush vulnerability) can cause partial or full loss of operation via buffer overflow and/or resource exhaustion. In this way, the vulnerabilities taught by Oberheide are a minimum set of dependencies to operate the computer system because, if exploited, they cause loss of operability by a user.

227. Thus, Oberheide teaches or suggests constraining its analysis to specific subsets of vulnerabilities from the set of system vulnerabilities (*i.e.*, *constraint set*), including to vulnerabilities that would shut down parts or all of the computer system. This is similar to the ’385 Patent’s description of a “constraint set,” which includes a “subset” of the total vulnerabilities such that the “vulnerabilities in a constraint set represent the vulnerabilities required for some system functionality.” ’385 Patent

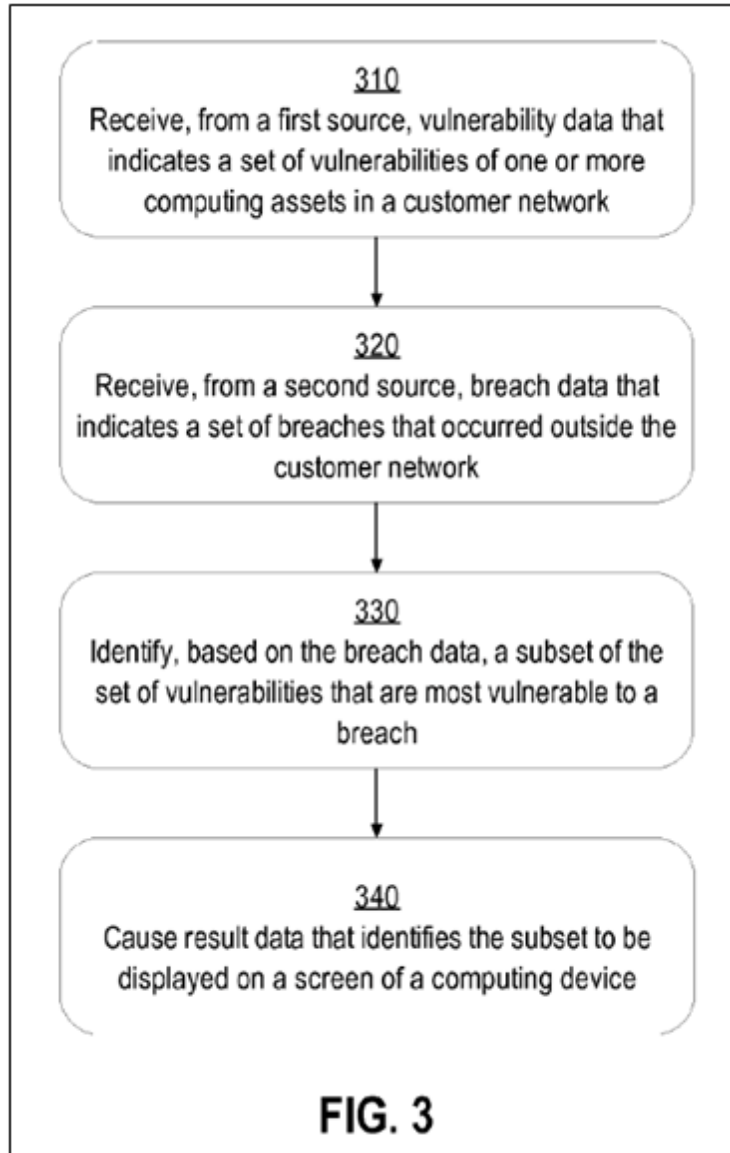
(EX1001), 3:29-32, 13:7-12.

228. Should Patent Owner or its expert argue that Oberheide does not teach an express process for “creating” this constraint set, Roytman teaches this.

229. Roytman teaches a “vulnerability threat management platform” that “rank[s] a set of vulnerabilities of a computing asset” and identifies “a subset of the set of vulnerabilities that are most vulnerable to the breach.” Roytman (EX1007), Abstract, [0025], [0032]-[0033], [0064]-[0068], Fig. 3. A “breach is a successful exploit[,]” that is, “a breach is a successful attack on a computing asset by successfully exploiting a vulnerability of the computing asset.” *Id.*, [0032].

230. To identify the subset of most exploitable vulnerabilities, Roytman’s threat management platform receives vulnerability data, receives breach data, correlates vulnerability and breach data, and then constructs a “ranked or ordered list of vulnerabilities.” *Id.*, [0025], [0033]. Roytman may “identify one or more of the most important vulnerabilities of a customer (based on breach data associated with the vulnerabilities). For example, the top five vulnerabilities with the highest number of breaches (or the vulnerabilities that are associated with at least a threshold number of breaches) in a most recent time interval are identified[.]” *Id.*, [0059].

231. An overview of this process is depicted below in Roytman’s Figure 3:



Id., Fig. 3.

232. Roytman explains that “FIG. 3 illustrates an example method for identifying a subset of vulnerabilities from a set of vulnerabilities that are most vulnerable to a breach based on breach data.” *Id.*, [0064]. “In step 330, a subset of the set vulnerabilities that are most vulnerable to a breach are identified based on the breach data received.” *Id.*, [0067], [0068].

233. **In sum**, the Oberheide-Roytman combination teaches “creating a constraint set of the computer system from the set of vulnerabilities [Oberheide constraining its vulnerability analysis to specific vulnerabilities pursuant to Roytman], the constraint set comprising a minimum set of dependencies to operate the computer system [Oberheide’s constrained vulnerabilities disrupt operations of some or all of a computer system].”

a. Motivation to Combine Oberheide with Roytman (Minimum Constraint Set)

234. It would have been obvious, and a POSITA would have been motivated to modify Oberheide to rank vulnerabilities as taught by Roytman for at least two reasons.

235. **First**, a POSITA would have been motivated to modify Oberheide to rank vulnerabilities correlated with an exploit pursuant to Roytman to create “a subset of the set of vulnerabilities that are most vulnerable to the breach.” Roytman’s ranking techniques would effectuate and improve Oberheide’s process to constrain its vulnerability analysis to specific vulnerabilities based on “enabled capabilities after the exploitation.” It would have therefore been obvious, and a POSITA would have been motivated to modify Oberheide to analyze and patch a subset of the highest ranked vulnerabilities ranked according to which vulnerabilities are most vulnerable (*i.e.*, amenable) to successful exploitation.

236. Ranking vulnerabilities based on a risk assessment was a common

practice. For instance, Felderer (EX1035) describes “Risk-based testing” (RBT) as an approach to “optimize” the management of risks in “mission-critical software” that “stem from software vulnerabilities and from defects in software functionalities[.]” Felderer (EX1035), 1-4. “RBT requires focusing the testing activities and efforts based on the risk assessment of the particular product[.]” *Id.* 6. Furthermore, the results of the RBT “may trigger additional measures to reduce either the likelihood or impact of a risk to occur in the field.” *Id.*, 7.

237. **Second**, using Roytman’s ranking system would have allowed Oberheide to prioritize vulnerabilities based on which components would be affected by the vulnerability. Roytman’s rankings include both risk score and optionally how important the component being exploited is. Roytman (EX1007), [0044] (ranking accounting for whether a particular computing asset is “absolutely critical ... for day-to-day operations”), [0050] (discussing “risk assessment unit”).

238. Next, Roytman suggests creating a subset of vulnerabilities with the highest rankings, including vulnerabilities with the highest risks or that would render inoperable critical components in the system and/or the system as a whole. Roytman’s subset of highest ranked vulnerabilities thus includes vulnerabilities associated with “a minimum set of dependencies to operate the computer system.” In other words, by distinguishing certain vulnerabilities based on the corresponding susceptible computer component, Roytman’s system creates a subset of

vulnerabilities (*i.e.*, constraint set) that are important to the operation of that critical component and therefore the computer system. Providing this as an additional factor for ranking vulnerabilities in Oberheide's system would have improved its prioritization of certain vulnerabilities for patching that would affect the most important components of the computer system that are necessary for operation of the computer system.

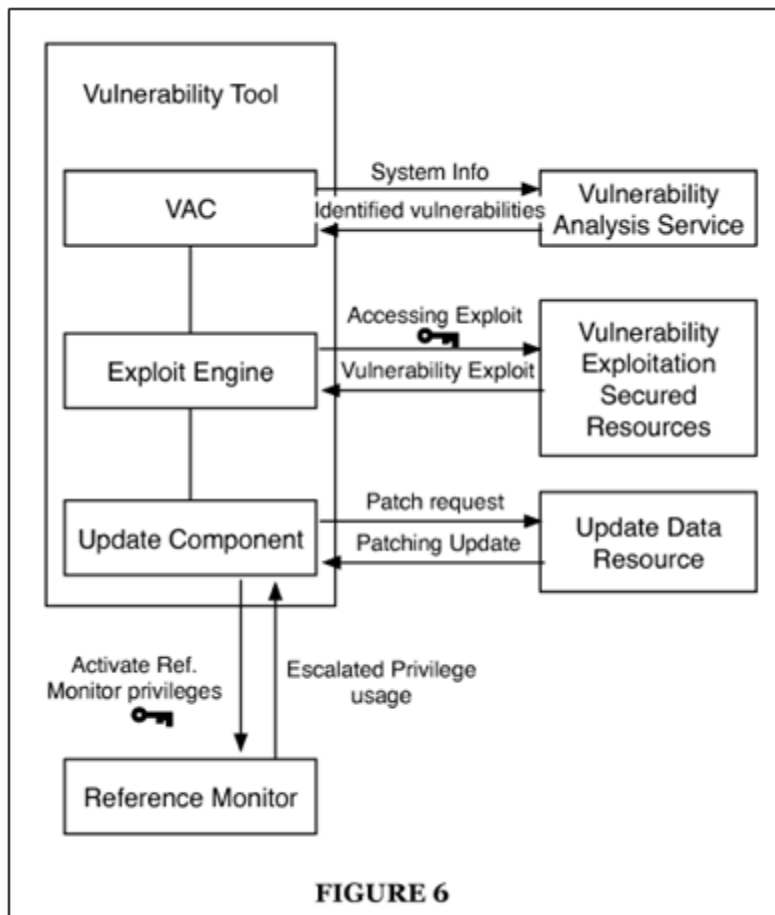
239. A POSITA would have had a reasonable expectation of success modifying Oberheide to rank vulnerabilities to create a subset of the set of vulnerabilities that are most vulnerable (*i.e.*, applicable) to the breach. Such a modification would have required simple programming changes well within the skill of a POSITA to a system that already constrains analysis to specific vulnerabilities. A POSITA would have recognized that modifying Oberheide's system, which already separates vulnerabilities from a group based on certain factors (*e.g.*, "robustness, simplicity, enabled capabilities after the exploitation"), to include a ranking of the vulnerabilities would have required minimal software changes. For example, it would have been an insignificant change to include more information (*e.g.*, risk of exploitation or associated vulnerable components) in an already established classification system for a more targeted classification based on those additional factors. *See also* §V.E above (discussing how creating a subset of vulnerabilities based on certain factors, such as whether they were essential for

critical assets, was a commonly known method).

240. **In sum**, Oberheide's system already distinguishes vulnerabilities based on whether they will affect a certain component or provide certain privileges once exploited, which, as modified by Roytman, ranks vulnerabilities that are most vulnerable to a breach (*i.e.*, "creating a constraint set of the computer system from the set of vulnerabilities, the constraint set comprising a minimum set of dependencies to operate the computer system").

12. Claim 1(h): applying the set of exploits to the constraint set of the computer system;

241. Oberheide's system applies exploits to the identified subset of vulnerabilities to determine which exploit will provide the necessary privileges for patching. Oberheide (EX1005), [0017]-[0019], [0024]-[0025]. This process is illustrated below in Oberheide's Figure 6:



Oberheide (EX1005), Fig. 6.

242. Namely, after Oberheide’s modified system identifies the constraint set of vulnerabilities, Oberheide teaches an “exploit engine” that “functions to execute an exploitation process.” *Id.*, [0017]. “The exploit engine preferably uses at least one of the vulnerabilities identified by the VAC to grant the vulnerability tool heightened permissions within the operating system” to facilitate a patch. *Id.*; see also *id.*, [0025] (“The exploit preferably grants the vulnerability tool the necessary privileges to patch the vulnerability”). To accomplish this, the “exploit engine may access a resource storing a collection of exploitation scripts or routines” such that an

“exploitation routine preferably targets at least one vulnerability ... identified by the analysis engine.” *Id.*; *see also id.*, [0019] (describing a “method for patching a device through exploitation”); [0024] (“Entering an operating mode of escalated privileges preferably includes selecting an appropriate vulnerability exploitation and executing the vulnerability exploitation”).

243. Taken together, Oberheide’s modified system selects sets of dark net exploits that will grant the vulnerability tool heightened (*i.e.*, elevated) permissions within the operating system to patch the vulnerabilities identified in its constraint set. Oberheide applies these sets of exploits against critical vulnerabilities in its constraint set to patch those vulnerabilities (*i.e.*, “applying the set of exploits to the constraint set of the computer system”).

244. **In sum**, Oberheide teaches “applying the set of exploits [Oberheide executing an exploitation process] to the constraint set of the computer system [Oberheide executing against the identified vulnerabilities].”

13. *Claim 1(i): analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and*

245. Oberheide’s modified system teaches analyzing the application of the set of exploits on the computer system to detect a particular vulnerability to patch, such as the highest ranked vulnerability. *See* §§VII.A.11-VII.A.12, Claims 1(g)-1(h); Oberheide (EX1005), [0024]-[0025].

246. Oberheide exploits the vulnerabilities in its constraint set to gain access to privileges necessary to patch those vulnerabilities. *See* §VII.A.12 above, Claim 1(h). Oberheide’s system is therefore “*analyzing the application of the set of exploits on the computer system*” by analyzing which exploits provide the system with the requisite privileges for patching a specific vulnerability. As taught by Oberheide (EX1005), “[i]f a plurality of vulnerabilities were identified, the selected vulnerability for exploitation may be made based on robustness, simplicity, enabled capabilities after the exploitation, or other properties of the exploit.” Oberheide (EX1005), [0024]. In determining these properties of the vulnerabilities for making its selection, Oberheide’s system is “analyzing the application of the set of exploits on the computer system.”

247. As modified by Roytman, Oberheide’s system would analyze and apply exploits to identify the highest ranked vulnerability to patch. Roytman (EX1007), [0033] (“the vulnerability threat management platform may then provide a ranked or ordered list of vulnerabilities which represent the order in which the vulnerabilities should be addressed, such that the vulnerability that poses the most significant threat is addressed first and the one that poses the least significant threat is addressed last”). Oberheide’s system thus “*detect[s] a particular vulnerability of the computer system*” by analyzing the exploitation of those vulnerabilities to identify the highest ranked vulnerability for patching. A POSITA would have been

motivated to combine Oberheide and Roytman as discussed in §VII.A.11.a above.

248. **In sum**, Oberheide teaches “analyzing the application of the set of exploits on the computer system [Oberheide determines properties and outcomes of exploiting multiple vulnerabilities] to detect a particular vulnerability of the computer system [Roytman selecting a particular vulnerability based on its ranking].”

14. Claim 1(j): altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.

249. Oberheide’s vulnerability tool includes an “update component” that applies “a patch or system update to resolve or reduce an existing vulnerability.” Oberheide (EX1005), [0018], [0019]. A “patch” or “system update” alters a configuration of the computer system. For example, as claimed and described by the ’385 Patent (EX1001) one manner of “altering a configuration of the computer system” includes “installing a patch to a software component.” ’385 Patent (EX1001), Claim 14. Oberheide’s application of a “patch” is therefore consistent with the ’385 Patent’s description that includes “altering a configuration of the computer system.”

250. The computer system is altered in response to the analysis of the application of the set of exploits. Oberheide teaches the applied exploits “grant[] the vulnerability tool the necessary privileges to patch the vulnerability.” Oberheide

(EX1005), [0025]. For instance, after applying exploits, Oberheide's system enters an escalated privileges mode and updates the system with a "vulnerability resolution" to resolve or reduce vulnerabilities. *Id.*, [0026]. "Updating the system preferably includes downloading, accessing, and/or selecting a patch or software update that can be applied to the system." *Id.* And "at least the vulnerability that is exploited is patched." *Id.* Fixing, updating, or reducing vulnerabilities reduces the potential damage of a cyberattack. Said another way, reducing the total number of vulnerabilities on a computer system reduces the potential damage of a cyberattack by disrupting and/or blocking the exploit's attack path and limiting the overall damage. *See also* §V.G above (discussing how patching vulnerabilities was a well-known method for improving cybersecurity).

251. **In sum**, Oberheide teaches "altering a configuration of the computer system [Oberheide patching vulnerabilities] in response to the analysis of the application of the set of exploits [Oberheide's analysis on the ranked vulnerabilities by Roytman] to reduce potential damage of a cyberattack [Oberheide's desire to prevent future exploitation]."

15. *Claim 8(Pre): A method for improving a computer device, the method comprising:*

252. To the extent the preamble is limiting, Oberheide teaches "a method for improving a computer device." *See* §VII.A.4 above, Claim 1(Pre); Oberheide (EX1005), [0019] (disclosing "a **method** for patching a device"); Oberheide

(EX1005), claims 1, 11, and 19. Protecting a computer device from vulnerabilities improves a computer device.

16. Claim 8(a): “accessing data comprising dark net information associated with a computer system;”

253. The Oberheide-Patsos combination teaches “accessing data comprising dark net information associated with a computer system.” See §VII.A.8 above, Claim 1(d); Patsos (EX1006), 295-296 (describing collecting security information related to exploits from underground web sources such as “hacking web sites”).

17. Claim 8(b): obtaining a set of exploits from the dark net information, the set of exploits configured to bypass a security feature of the computer system;

254. As an initial matter, a POSITA would have understood that there is no meaningful difference between Claim 8(b)’s “bypass” and Claim 1(e) “penetrate”: both refer to exploits that circumvent security features or break into computer systems and software.

255. The Oberheide-Patsos combination teaches obtaining a set of exploits from the dark net information using a dark net crawler. See §VII.A.9 above, Claim 1(e). These dark net exploits are “configured to bypass a security feature of the computer system” to “grant[] the vulnerability tool the necessary privileges to patch the vulnerability.” Oberheide (EX1005), [0024]-[0025]. A POSITA would have known that there was no meaningful difference between “penetrat[ing] a computer system” and “bypass[ing] a security feature of the computer system.” Therefore, the

descriptions above apply equally to this limitation. *See* §VII.A.9 above, Claim 1(e). (explaining how Oberheide teaches entering an operating mode with escalated privileges via exploits (*i.e.*, penetrating the computer system)).

18. Claim 8(c): applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;

256. The Oberheide-Patsos combination teaches applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities. *See* §VII.A.10 above, Claim 1(f).

19. Claim 8(d)(1): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system,

257. The Oberheide-Patsos-Roytman combination teaches “creating a constraint set of vulnerabilities of the computer system” by ranking and constraining its vulnerability analysis to specific vulnerabilities. *See* §VII.A.11 above, Claim 1(g). This “constraint set” comprises “a minimum set of dependencies to operate the computer system” such that Oberheide-Patsos-Roytman’s vulnerability analysis may be constrained to vulnerabilities that shut down the computer system through successful exploitation and/or to vulnerabilities that render facets of the computer system inoperable. *See* §VII.A.11 above, Claim 1(g).

20. Claim 8(d)(2): wherein application of the set of exploits on the computer system comprises determining the effect of the set of exploits on the constraint set of vulnerabilities of the computer system;

258. Oberheide’s system applies exploits to the identified subset of

vulnerabilities to determine which exploit will provide the necessary privileges for patching. *See* §VII.A.12 above, Claim 1(h); Oberheide (EX1005), [0017]-[0019], [0024]-[0025]. Oberheide determines the effect of the set of exploits on the constraint set of vulnerabilities of the computer system by determining which exploit provides privileges for patching. *Id.*

21. *Claim 8(e): analyzing an application associated with the set of exploits on the computer system to detect a particular vulnerability of the constraint set of vulnerabilities of the computer system; and*

259. Oberheide's modified system teaching analyzing the application of the set of exploits on the computer system to detect a particular vulnerability to patch, such as the highest ranked vulnerability. *See* §VII.A.13 above, Claim 1(i).

22. *Claim 8(f): altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.*

260. Oberheide's vulnerability tool includes an "update component" that applies "a patch or system update to resolve or reduce an existing vulnerability" such that fixing, updating, or reducing vulnerabilities reduces the potential damage of a cyberattack. *See* §VII.A.14 above, Claim 1(j); Oberheide (EX1005), [0018], [0019].

23. *Claim 14: "The method of claim 8, wherein altering the configuration of the computer system comprises installing a patch to a software component of the computer system that is exposed by the set of exploits to attack the computer system."*

261. As in §VII.A.14, Oberheide installs patches to a software component of the computer system which *alter the configuration of the computer system*. *See*

§VI.A.20 above, Claim 8(f), §VII.A.14 above, Claim 1(j). Oberheide teaches installing patches for the vulnerable component that was exploited. Oberheide (EX1005), [0026] (“us[ing] the privileges granted through the exploit to resolve or reduce vulnerabilities,” and “**at least the vulnerability that is exploited is patched**”). Particularly, “[u]pdating the system preferably includes **downloading, accessing, and/or selecting a patch or software update** that can be applied to the system. The updating can target only the identified vulnerability.” Oberheide (EX1005), [0026].

262. **In sum**, Oberheide teaches “wherein altering the configuration of the computer system comprises installing a patch to a software component of the computer system that is exposed by the set of exploits to attack the computer system [Oberheide patching at least the vulnerability that is exploited].”

24. Claim 15(Pre): “One or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine, the computer process comprising:”

263. To the extent the preamble is limiting, Oberheide teaches one or more non-transitory tangible computer-readable storage media storing computer-executable instructions for performing a computer process on a machine. See §VII.A.7 above, Claim 1(c).

25. Claim 15(a): “obtaining a set of exploits from dark net information, the set of exploits configured to bypass a security feature of a computer system;”

264. See §VII.A.9 above, Claim 1(e), and §VII.A.17 above, Claim 8(b) (identical language).

26. Claim 15(b): “applying an exploit function which takes the set of exploits as input and returns a set of vulnerabilities;”

265. See §VII.A.10 above, Claim 1(f), and §VII.A.18 above, Claim 8(c) (identical language).

27. Claim 15(c): “creating a constraint set of vulnerabilities of the computer system from the set of vulnerabilities comprising a minimum set of dependencies to operate the computer system;”

266. See §VII.A.19 above, Claim 8(d)(1) (identical language), and VII.A.11 above, Claim 1(g).

28. Claim 15(d): “applying the set of exploits to the constraint set of vulnerabilities of the computer system;”

267. See §VII.A.12 above, Claim 1(h).

29. Claim 15(e): “analyzing the application of the set of exploits on the computer system to detect a particular vulnerability of the computer system; and”

268. Oberheide’s modified system teaches analyzing the application of the set of exploits on the computer system to detect a particular vulnerability to patch, such as the highest ranked vulnerability. See §VII.A.13 above, Claim 1(i), and §VII.A.21 above, Claim 8(e).

30. Claim 15(f): “altering a configuration of the computer system in response to the analysis of the application of the set of exploits to reduce potential damage of a cyberattack.”

269. See §VII.A.14 above, Claim 1(j), and §VII.A.22 above, Claim 8(f)

(identical language).

VIII. OPINION – GROUND 3

270. Claims 2-6, 9-12, and 16-17 would have been obvious over Oberheide (EX1005), Patsos (EX1006), Roytman (EX1007), and Juels (EX1008).

A. Obvious to Combine Oberheide, Patsos, Roytman, and Juels

1. Overview of Juels

271. U.S. Patent No. 9,471,777 to Juels *et al.* (“Juels”) was filed on February 24, 2012 (EX1008).

272. Juels teaches using a game-theoretic model to protect a processing system from security threats. Juels (EX1008), 1:7-10, 2:7-10. The model may include both “attacker and defender [that] entities can take actions at any time, with the actions having associated costs.” *Id.*, 2:34-38. The processing system identifies “a plurality of defensive security actions to be taken to address a persistent security threat to a system comprising information technology infrastructure.” *Id.*, 2:19-22. Optimal defensive security actions are determined based on the actions taken in the game-theoretic model, such as “key update operations, virtual machine refresh operations, cloud service auditing operations, password reset operations, and many others.” *Id.*, 2:41-43, 15:24-26 (describing password reset based on attackers purchasing passwords in an “underground forum”).

273. Juels is in the same field of endeavor as the ’385 Patent because both are directed to systems and methods for exploiting vulnerabilities and patching

vulnerabilities. *Compare* '385 Patent (EX1001), 3:4-15, 3:61-64, 13:12-18, Figs. 5 and 7 *with* Juels (EX1008), 2:39-43, 12:60-63 (“A move by the defender may be a system-wide remediation campaign, e.g., patching of all machines in a network”), 13:6-11 (“The goal of the attacker is to compromise the [computing] device by exploiting a software vulnerability or credential compromise.”), 13:25-34.

274. The '385 Patent describes common problems in the art related to improving the security of a computer system or device. '385 Patent (EX1001) 2:63–3:3 (describing the defender of a target computer system wanting to “minimize the effectiveness of an attacker”), 13:15-18. Juels similarly addresses systems and methods for improving the security of a computer system or device. Juels (EX1008), Abstract, 1:7-10 (the invention relates “particularly to techniques for protecting such [information processing] systems from persistent security threats.”), 2:7-10 (addressing “defensive security actions in information processing systems”), 2:39-47. As such, Juels is reasonably pertinent to the problems of the '385 Patent and is analogous art.

2. *Claim 2: The system of claim 1, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.*

275. The Oberheide-Patsos-Roytman combination analyzes and applies exploits against the highest ranked vulnerabilities. *See* §VII.A.13 above, Ground 2, Claim 1(i). To rank vulnerabilities, Roytman teaches that its “vulnerability threat

management platform may provide a quantified measure, known as a risk score herein, to indicate how vulnerable a particular asset may be of being successfully exploited or breached.” Roytman (EX1007), [0034].

276. For the reasons explained in §VII.A.13 above it would have been obvious to implement Roytman’s ranking process, including calculating risk scores, into Oberheide’s system. *See* §VII.A.13 above, Claim 1(i). However, Oberheide’s modified system does not expressly quantify risk level related to a cost to obtain exploits. It would have been obvious to include a cost metric for exploits as taught by Juels (EX1008), when generating a risk score in Oberheide’s modified system.

277. Juels teaches a game-theoretic model where an attacker acts to compromise a target computing device, and a defender acts to protect the target device. Juels (EX1008), 2:32-38, 13:6-15. Because “[a]n action by either side carries a cost,” the actions taken must be cost-effective. *Id.*, 13:12-24. In one example, the attacker chooses between (1) a less expensive, yet less effective published exploit; or (2) a more expensive, yet more effective zero-day exploit. *Id.*, 13:25-29. A POSITA would have recognized, as taught by Jules, that the well-known link between the effectiveness of the exploit (*i.e.*, “the risk level posed to the computer system”) is “related to the cost to obtain the set of exploits”. For example, a POSITA would have understood that a more expensive exploit is less likely to be purchased by a hacker than a less expensive one. This would have inherently

affected the risk level posed to a computer system because a cost-prohibitive exploit would be less likely to be purchased and used (*i.e.*, lower risk level).

278. **In sum**, the Oberheide-Patsos-Roytman-Juels combination, teaches “wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system [Roytman quantifying a risk score when ranking vulnerabilities] related to a cost to obtain the set of exploits [Juels including the cost of the exploit in its calculation of the risk score].”

a. Motivation to Combine Oberheide with Juels (Exploit Costs)

279. It would have been obvious, and a POSITA would have been motivated to further modify Oberheide to consider exploit cost, as taught by Juels (EX1008), when generating a risk score. Juels (EX1008), 13:21-34 (teaching zero-day exploits cost more but are more effective than less-expensive published exploits).

280. A POSITA would have recognized that including the cost in its ranking determination would have more precisely ranked the riskiest vulnerabilities to patch first. For example, in the modified system, a vulnerability may have been initially ranked the highest priority for patching because it was found critical for computer operation. However, the modified system would have additionally determined that the exploit for that vulnerability was being sold at a cost-prohibitive price point. As such, the system may rank that vulnerability lower on its list, depending on the cost

of the exploits targeting the next-ranked vulnerabilities. This example illustrates how the modification would increase efficiency of identifying, exploiting, and patching the highest priority vulnerabilities in Oberheide's system.

281. In other words, the modified system would distinguish a more effective, but cost-prohibitive exploit, from a slightly less effective, but cheaper exploit. This would enhance selectivity of the vulnerabilities to be patched, leading to an overall enhanced security process.

282. The above example considers an optimization problem across three dimensions: cost vs. risk level vs. exploit effectiveness. This risk-reward analysis is well known to even lay persons, who often engage in even more complex, multi-dimensional tradeoff analyses. For example, consider car insurance: one often assesses the (1) cost of higher and higher car insurance premiums and coverage options vs. (2) the cost to repair one's vehicle involved in an accident vs. (3) one's driving habits and history (*e.g.*, risky vs. safer drivers) vs. (4) the locality's prevalence for accidents (*e.g.*, urban vs. rural municipalities).

283. A POSITA would have appreciated a reasonable likelihood of success modifying the Oberheide-Patsos-Roytman combination to generate a risk score for a vulnerability that accounts for the cost to exploit that vulnerability. Roytman's risk score already accounts for how likely the vulnerability may be exploited and how important is the system component that could be exploited. A POSITA would

have therefore understood that the system was well-equipped to analyze particular metrics of a vulnerability to provide a final ranking. As such, modifying this risk score to account for a third metric, cost to exploit, would be well within the skill of a POSITA at the time. Such a modification would have required simple software modifications to include and weigh the cost of the exploit in addition to the other metrics in determining its risk score.

284. **In sum**, Oberheide’s modified system already ranks vulnerabilities and applies a risk score as taught by Roytman, which, as modified by Juels, accounts for the cost of the exploit for exploiting that vulnerability (*i.e.*, “wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits”).

3. ***Claim 3: The system of claim 2, wherein the processing device further performs the operation of: aligning Internet data with the constraint set of vulnerabilities based on information obtained about the set of exploits.***

285. As an initial matter, the ’385 Patent does not describe “aligning Internet data” outside of Claim 3. Given this, as understood in light of the specification, a POSITA would have understood “aligning Internet data” to at least include correlating Internet data with the vulnerabilities based on information obtained about the set of exploits.

286. As discussed in §VII.A.11 above, the Oberheide-Patsos-Roytman

combination teaches creating a “constraint set” by constraining its vulnerability analysis to specific vulnerabilities, including by ranking vulnerabilities, as taught by Roytman (EX1007), to identify a subset of vulnerabilities most vulnerable (*i.e.*, amenable) to exploitation. *See* §VII.A.11 above, Ground 2, Claim 1(g). Roytman ranks vulnerabilities in part by receiving “breach data, sourced from across the internet, and correlat[ing] that data with vulnerabilities discovered in computing assets.” Roytman (EX1007), [0030], [0032] (“a breach is a successful exploit”).

287. Breach data “indicates a set of breaches that occurred outside the customer network” and is used to determine “the likelihood that an exploit of a particular vulnerability of a particular asset will be successful.” *Id.*, [0025], [0030]. Breach data therefore includes “Internet data” that is “based on information obtained about the set of exploits” (*i.e.*, breaches), and is aligned (*e.g.*, correlated) with the vulnerabilities such that breach data is used to determine, in part, the vulnerability’s rank.

288. A POSITA would have been motivated to modify and would have appreciated a reasonable expectation of success in modifying Oberheide with Roytman for the same reasons above. *See* §VII.A.8 above, Ground 2, Claim 1(g).

289. **In sum**, the Oberheide-Patsos-Roytman-Juels combination teaches “wherein the processing device further performs the operation of: aligning Internet data with the constraint set of vulnerabilities [Roytman correlating breach data with

vulnerabilities] based on information obtained about the set of exploits [Roytman's information obtained about breaches].”

4. Claim 4: The system of claim 3, wherein obtaining a set of exploits through the network connection comprises: applying one or more algorithms to obtain the set of exploits.

290. As an initial matter, the term “algorithm” in this claim appears to be a nonce and no different than saying a “technique,” because any set of concrete steps can be considered an algorithm. Computer algorithms can span from complex (*e.g.*, neural network AI) to trivial (*e.g.*, multiplying two numbers).

291. Oberheide's modified system obtains exploits through a network connection using, *e.g.*, a dark net crawler. *See* §VII.A.8 above, Ground 2, Claim 1(d) (describing Patsos's techniques for obtaining exploit code from underground websites). An automated crawler includes one or more algorithms for obtaining exploits from various dark net websites. *See, e.g.*, Albanese (EX1036), 2 (describing how “[a]lgorithms exist to find the set of exploits from which the goal conditions are reachable.”).

292. Internet crawlers operate algorithmically: they require a defined set of logical instructions to systematically navigate web content and extract data. *See*, §V.A above. For example, a classical web crawler begins by collecting a set of web sites. For each web site, it uses HTTP to connect to the web server and download the HTML content, often in a file named “index.html.” The crawler then parses that

index.html file looking for hyper-references (HREFs) to other web pages of the same or other web sites. The crawler then adds those web pages to the list of additional web pages to crawl next. Each of those web pages may in turn HREF to yet more web pages, and so on. Thus, a web crawler has to recursively scan a rapidly growing set of web pages (while avoiding loops as web pages can self-reference or form cycles). Indeed, it was a then small company named Google who developed massive, automated, parallel, algorithmic web crawling and indexing in the late 1990s, which I observed personally.

293. Moreover, dark net crawling algorithms were well known in the art. For example, Olston describes “deep web crawlers” that “automatically fill in forms to reach the content behind them.” Olston (EX1015), 230; *see also*, Fu (EX1018), 1218 (describing development of “a Dark Web forum spidering system” that includes “language-independent URL ordering features to allow spidering of Dark Web forums across languages,” “a rule-based URL ordering technique couples with BFS and DFS crawl space traversal ... to minimize the amount of irrelevant Web pages collected,” and “utilizing an incremental crawler that uses forum wrappers to determine the subset of threads that need to be collected.”). A POSITA would have understood Oberheide’s modified system would have implemented known algorithms to crawl the dark net and obtain exploits as taught by Patsos. Said another way, the systems employed in Oberheide’s modified system that crawl the dark net

for obtaining exploits would have necessarily used algorithms such as those taught and/or suggested by Patsos.

294. **In sum**, the Oberheide-Patsos-Roytman-Juels combination teaches “wherein obtaining a set of exploits through the network connection comprises: applying one or more algorithms to obtain the set of exploits [Patsos using the dark net crawlers that operate algorithmically to obtain the exploit code].”

5. ***Claim 5: The system of claim 4, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtains the set of exploits constrained by the determined cost to obtain the set of exploits.***

295. Oberheide’s modified system benefits from updating its local repository of exploit scripts with exploits from the dark net. *See* §VII.A.8 above, Ground 2, Claim 1(d). Moreover, Oberheide’s modified system determines the cost to obtain the set of exploits, as taught by Juels (EX1008), for ranking vulnerabilities (*e.g.*, by likelihood of exploitation). *See* §VIII.A.2 above, Claim 2. Cost information would have been collected by the dark net crawler (*i.e.*, one or more algorithms) while obtaining the set of exploits. *See* §VII.A.9 above, Claim 1(e), and §VIII.A.2 above, Claim 2; Nielsen (EX1037), 3:58-63, 6:14-29 (teaching a web crawler capable of searching for cost information related to service plans, the web crawler comprising a range checker for comparing that cost information to a specific price range). As such, Oberheide would have similarly benefitted from buying exploits based on collected cost information to update its repository and creating a

budget to regulate this system.

296. It would have been obvious to create a budget and obtain the set of exploits constrained by the determined cost to obtain the set of exploits. Juels's "Host Takeover" version of its game-theoretic model involves an attacker trying to "compromise [a] device by exploiting a software vulnerability." Juels (EX1008), 13:5-34. The compromises available to the attacker for exploitation have "varying costs," such as a "zero-day exploit" costing more than "a published exploit," forcing the players to "choose among a set of actions with varying costs and effectiveness." *Id.*

297. Juels describes that the players have certain "[b]udgets" they can't exceed, providing an "upper bounds ... on the players' respective move rates." *Id.*, 36:28-33. For example, a player may be limited in their play rate by a "financial budget." *Id.*, 37:4-9. In other words, Juels's model links the cost of an exploit to a financial budget that constrains the player's ability to obtain exploits.

298. **In sum**, the Oberheide-Patsos-Roytman-Juels combination teaches "wherein the one or more algorithms determine the cost to obtain the set of exploits [Juels collecting cost data while crawling the dark net, using Patsos's dark net crawler in combination with Juels's teachings on cost information] to create a budget [Juels's limiting the system using budgets that cannot be exceeded] and obtains the set of exploits constrained by the determined cost to obtain the set of exploits [Juels's

system creating an upper bound of costs for obtaining the exploits].”

a. Motivation to Combine Oberheide with Juels (Exploit Budgets)

299. A POSITA would have found it obvious and have been motivated to modify Oberheide to create a budget for obtaining exploits that is constrained by the cost of the exploits, as taught by Juels.

300. A POSITA would have understood, based on Juels’s teachings, that for obtaining costly exploits in Oberheide’s modified system, it would be beneficial to have a budget constraining the system from purchasing all exploits it found. A POSITA would have recognized that without a specified budget, the system would suffer from runaway purchasing power, spending an unforeseen amount on updating exploits in its repository. For instance, without constraint based on a determined total amount willing to spend on exploits, the algorithms responsible for crawling and obtaining exploits may obtain every exploit with the intended goal of maximizing security.

301. Conversely, a POSITA would have understood that (1) Oberheide’s modified system would not be operating with unlimited funds, and (2) that the cost to obtain some exploits would not be worth it. Regarding the latter, a POSITA would have recognized that, similar to Oberheide’s ranking system, some exploits may be expensive but have minimal to no risk of being used. As such, a POSITA would have recognized that such an exploit would not be worth purchasing to update

Oberheide's exploit database.

302. Moreover, by constraining the system by cost, Oberheide would only purchase exploits it knew would substantially benefit the system. For example, in ranking exploits, Oberheide may identify two top exploits providing nearly the same privilege level, or risk level, required for patching a subset of vulnerabilities. However, Oberheide's system, including cost, determines that the top-ranked exploit was much more expensive than the second-ranked exploit. In this example, Oberheide's system would benefit by purchasing and reranking the second-ranked exploit over the top-ranked exploit.

303. A POSITA would have had a reasonable likelihood of success in modifying Oberheide in this way for the same reasons as discussed in §VIII.A.2.a above, Claim 2.

304. **In sum**, Oberheide's modified system would have included collecting the cost information and applying a budget to the algorithm for acquiring new exploits for updating its exploit engine (*i.e.*, one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtain the set of exploits constrained by the determined cost to obtain the set of exploits).

6. ***Claim 6: The system of claim 5, wherein obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.***

305. Oberheide's system, as modified by Juels, teaches obtaining the set of

exploits based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits. *See* §VIII.A.5 above, Claim 5 (providing example of maximizing the quantifying of the risk level of two exploits as related to the cost to obtain the set of exploits).

306. More specifically, the Oberheide-Patsos-Roytman-Juels combination teaches quantifying a risk level posed to the computer system related to the cost to obtain the set of exploits. *See* §VIII.A.2 above, Claim 2. For example, the risk level “indicates how vulnerable a particular asset may be of being successfully exploited or breached.” Roytman (EX1007), [0069]. “[T]he risk score may represent a value between 0 and 1000, where the **higher the number, the more at risk that particular computing asset** or group of assets is of being successfully exploited or breached.” Roytman (EX1007), [0034].

307. The attacker in Juels’s game-theoretic model is best able to exploit a software vulnerability by obtaining exploits that will be “more effective” against the targeted vulnerability. Juels (EX1008), 13:6-15, 21-34. Thus, quantifying the risk level is “maximized” by obtaining exploits, as permitted by cost, that may most effectively exploit the computer system.

308. As discussed in Claim 2, a POSITA would have found it obvious to modify Oberheide, Roytman, and Juels. *See* §VIII.A.2.a above, Claim 2. The “more effective” an exploit is against a targeted vulnerability, such as a computer system

or asset, the more vulnerable that target is to being “successfully exploited or breached” by that exploit. Juels (EX1008), 13:25-29 (differentiating a less effective “published exploit” from a “more effective” “zero-day exploit”); Roytman (EX1007), [0034] (teaching “provid[ing] a quantified measure, known as a risk score ... to indicate how vulnerable a particular asset may be of being successfully exploited or breached”).

309. This, in turn, means that a higher risk level is assigned to that targeted vulnerability. Allodi (EX1038), 1:6 (describing a risk score as a product of “Impact x Exploitability,” wherein “[t]he Impact submetric is an assessment of the impact the exploitation of the vulnerability has on the system.”); *see also* Giakouminakis (EX1039), [0027] (describing that a vulnerability in a computer system may have a “High” risk level “due to the ease and effectiveness of the exploit”).

310. As further discussed above in Claims 2 and 5, however, obtaining exploits from the dark net comes at a cost, and the budget is not limitless. *See* Claims 2 (§VIII.A.2 above) and 5 (§VIII.A.5 above). With that backdrop, a POSITA would have understood that “*maximizing [...] the risk level posed to the computer system*” is limited by the effectiveness of the exploits the POSITA may obtain based on the exploit budget. Gianini (EX1040), 1 (discussing maximizing an attack strategy while also considering “the finite resources available to the hackers (buying and massively deploying malware is costly)”).

311. A POSITA would have appreciated a reasonable expectation of success further modifying Oberheide's modified system to maximize the risk level versus costs. Roytman's "risk score" is determined "based on several contextual factors." Roytman (EX1007), [0070]. Roytman's factors are non-exhaustive, but include "the prevalence (or number) of exploits available for each vulnerability, the popularity of the computing asset or [...] difficulty of exploiting the vulnerability[.]" *Id.*

312. It would have required simple programming changes to consider exploit cost as a factor. *See, e.g.,* Frei (EX1041), 35 (teaching the incorporation of multiple "[f]actors that determine the market price of a vulnerability", including "exclusivity of information," "security impact," and "product popularity"). This cost may limit the number and/or type of exploits purchased, as well as the effectiveness of the exploits available based on the budget. A POSITA under these constraints would also understand that these factors directly impact the risk level assigned to the target computer system. *See* §VIII.A.2 above, Claim 2.

313. **In sum**, the Oberheide-Patsos-Roytman-Juels combination teaches "wherein obtaining the set of exploits [via Patsos's dark web crawler] is based on maximizing the quantifying of the risk level [Roytman ranking the identified exploits on the dark web] posed to the computer system related to the cost to obtain the set of exploits [Juels collecting the cost and including that in the calculations of the risk level, as taught by Juels]."

7. ***Claim 9: The method of claim 8, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.***

314. Oberheide's modified system teaches quantifying a "risk score" related to a cost to obtain the exploits. *See* §VIII.A.2 above, Claim 2.

8. ***Claim 10: The method of claim 9, wherein obtaining a set of exploits comprises: applying one or more algorithms to obtain the set of exploits based on the constraint set of vulnerabilities of the computer system from one or more darknet marketplaces on the Internet.***

315. Oberheide's modified system teaches applying one or more algorithms to obtain the set of exploits. *See* §VIII.A.4 above, Claim 4. To the extent Claim 10 is interpreted to require that the constraint set of vulnerabilities of the computer system to be determined before obtaining the set of exploits, Oberheide teaches this. Oberheide teaches that "vulnerabilities are identified ... for a particular component or set of components" (*i.e.*, constraint set of vulnerabilities). Oberheide (EX1005), [0020].

316. Oberheide's method then "select[s] an appropriate vulnerability exploitation and execut[es] the vulnerability exploitation" to gain "escalated privileges" through the vulnerability to "fix or lessen the threat." *Id.*; *see also, id.* (teaching a situation with multiple vulnerabilities where an exploit is selected from a group of exploits "based on the ... enabled capabilities after the exploitation."). Oberheide therefore obtains a set of exploits from its repository based on the

constraint set of vulnerabilities of the computer system.

317. As modified by Patsos, Oberheide obtains the vulnerabilities “from one or more darknet marketplaces on the Internet.” *See* §VII.A.9 above, Ground 2, Claim 1(e). To the extent that Patent Owner alleges that a “hacking web site” providing “exploits (and actual exploit code)” is not a darknet marketplace, a POSITA would have found it obvious to modify Oberheide to obtain exploits based on cost from a dark web “marketplace.” *See* §VIII.A.5 above, Claim 5; *see also* §V.B above.

318. Indeed, darknet marketplaces were widely known repositories for exploit data with well-understood algorithms available in the industry to obtain exploits from those marketplaces. *See* Kumar (EX1042) (“A new Dark Web Market, called ‘*TheRealDeal*,’ has opened up for hackers, which focuses on *selling Zero-Day exploits*[.]”); *see also* International Institute of Cyber Security (EX1043) (listing The RealDeal Market amongst a “list of some websites/market places for technology related stuff like zero day vulnerabilities, exploits, malwares and hackers etc.”).

319. **In sum**, the Oberheide-Patsos-Roytman-Juels combination teaches “wherein obtaining a set of exploits through the network connection comprises: applying one or more algorithms to obtain the set of exploits [Patsos’s dark net web crawler, algorithms to obtain exploits] based on the constraint set of vulnerabilities of the computer system [Oberheide’s specific vulnerabilities based on, *e.g.*, which

components they will affect] from one or more darknet marketplaces on the Internet [Patsos's hacking web sites].”

9. ***Claim 11: The method of claim 10, wherein the one or more algorithms determine the cost to obtain the set of exploits to create a budget and obtains the set of exploits constrained by the determined cost to obtain the set of exploits.***

320. Oberheide's modified system teaches crawling exploits from the dark net based on risk and cost and therefore teaches the method of Claim 11. *See* §VIII.A.5 above, Claim 5.

10. ***Claim 12: The method of claim 11, where obtaining the set of exploits is based on maximizing the quantifying of the risk level posed to the computer system related to the cost to obtain the set of exploits.***

321. Oberheide's modified system teaches the method of Claim 12. *See* §VIII.A.6 above, Claim 6.

11. ***Claim 16: The one or more non-transitory computer-readable storage media of claim 15, wherein analyzing the application of the set of exploits on the computer system comprises quantifying a risk level posed to the computer system related to a cost to obtain the set of exploits.***

322. Oberheide's modified system teaches one or more non-transitory computer-readable storage media storing instructions for performing this process. *See* §VIII.A.2 above, Claim 2.

12. ***Claim 17: The one or more non-transitory computer-readable storage media of claim 15, wherein obtaining a set of exploits comprises applying one or more algorithms to obtain the set of exploits from one or more darknet marketplaces on the Internet.***

323. Oberheide’s modified system teaches applying one or more algorithms to obtain the set of exploits from one or more darknet marketplaces on the Internet. *See* §VII.A.9 above, Ground 2, Claim 1(e), and §VIII.A.8 above, Claim 10.

IX. OPINION – GROUND 4

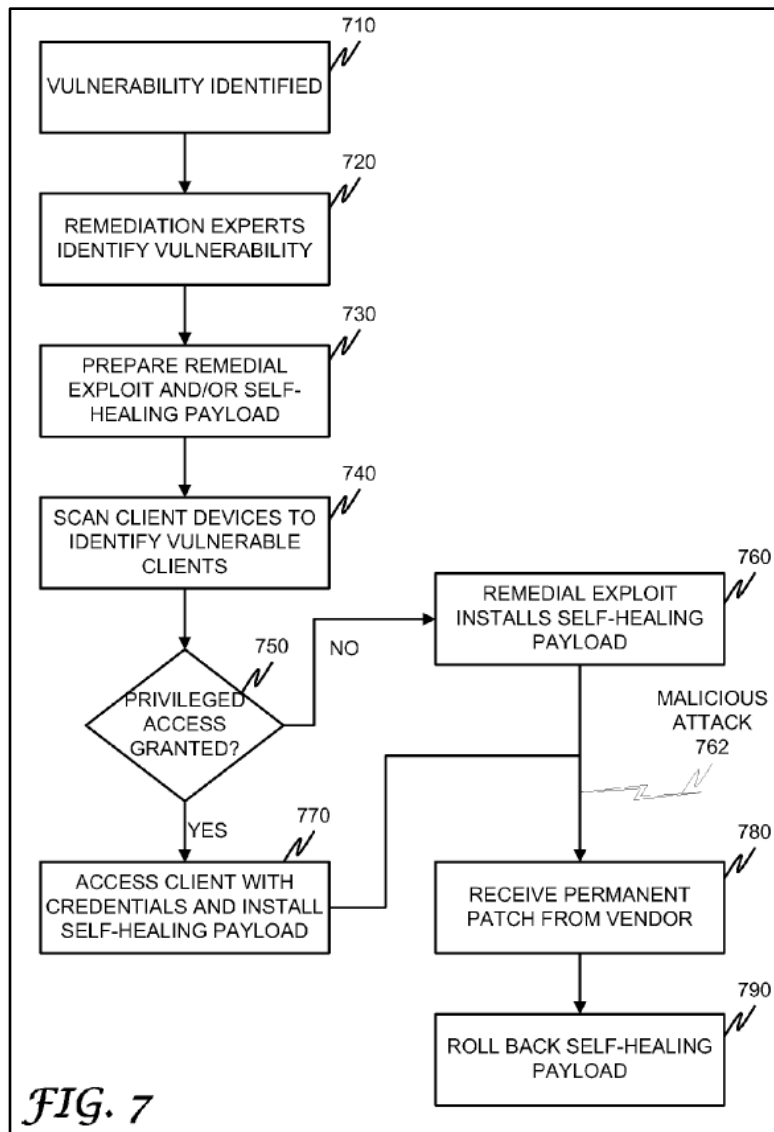
324. Claims 7 and 13 would have been obvious over Oberheide (EX1005), Patsos (EX1006), Roytman (EX1007), and Wolff (EX1009).

A. Obvious to Combine Oberheide, Patsos, Roytman and Wolff

1. Overview of Wolff

325. U.S. Patent Publication No. 2013/0198848 to Wolff (“Wolff”) was filed on October 17, 2011, and published on August 1, 2013 (EX1009).

326. Wolff teaches an approach to remediating “security vulnerabilities in computer software.” *Id.*, [0015]. Wolff describes a “computer security vulnerability remediation system (CSVRS)” that includes a CSVRS client coupled to a remediation server over a network, the CSVRS being a “computer or computing device.” *Id.*, [0014], [0029]. “The CSVRS client includes software having a security vulnerability” that may be exploited by a malicious actor.” *Id.* An exemplary method for this vulnerability remediation is depicted below in Wolff’s Figure 7:



Wolff (EX1009), Fig. 7.

327. In Wolff’s method, a Remediation Service Provider (RSP), which operates the remediation server, receives information on an exploit that may attack the identified vulnerability in the CSVRS client and “may modify the exploit to create a remedial exploit, including a self-healing payload.” *Id.*, [0014], [0019] (describing buying exploits on sale on “black market ‘underground economy

servers”), [0027]. The self-healing payload is deployed on the CSVRS client to take actions such as “downloading programs, executing programs, downloading software updates, installing patches, removing software, modifying privileges,” and many others. *Id.*, [0028]. If the actions are adequate, the self-healing payload will ameliorate the effects of an incoming malicious attack on the CSVRS client. *Id.*, [0028], [0047].

328. Wolff is in the same field of endeavor as the ’385 Patent because both are directed to systems and methods for exploiting vulnerabilities and patching vulnerabilities. *Compare* ’385 Patent (EX1001) 3:4-15 (describing “penetration testing that is fed with real-world exploit market data”), 3:61-64 (describing exploiting a set of vulnerabilities), 13:12-18 (reverse engineering exploits “to develop a patch to the exposed vulnerability), Figs. 5 and 7 *with* Wolff (EX1009), Abstract (describing a “vulnerability remediation system” that includes delivery of a remedial exploit through a vulnerability), [0014]-[0016], [0028], [0033].

329. The ’385 Patent describes common problems in the art related to improving the security of a computer system or device. ’385 Patent (EX1001) 2:63–3:3 (describing the defender of a target computer system as wanting to “minimize the effectiveness of an attacker”), 13:15-18. Wolff similarly addresses systems and methods for improving the security of a computer system or device. Wolff (EX1009), Abstract, [0002] (teaching “a computer security vulnerability

remediation system”), [0014] (using a “remedial exploit, which carries a self-healing payload” for “prevent[ing] the vulnerability from being further exploited”), Claim 1 (teaching a “self-healing payload configured to take a remedial action **to reduce the client machine’s exposure to the security vulnerability**”). As such, Wolff is reasonably pertinent to the problems of the ’385 Patent and is analogous art.

2. Claim 7: “The system of claim 1, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.”

330. Oberheide teaches “altering the configuration of the computer system” by applying a patch or software update to a software component of the computer system that is utilized by the set of exploits to attack the computer system. *See* §VII.A.14 above, Ground 2, Claim 1(j); Oberheide (EX1005), [0026] (“[p]referably at least the vulnerability that is exploited is patched” and “patch[ing] **the vulnerable software component**”). While Oberheide’s system “enable[s] capabilities to fix, update, and/or reduce the vulnerabilities,” Oberheide does not expressly teach removal of the exploited software component during patching. Wolff, however, teaches removing the exploited software component.

331. Wolff teaches a method of remediating a client computer by applying a remedial exploit to a software vulnerability. Wolff (EX1009), [0014]-[0015], [0028]. The remedial exploit carries a self-healing payload, which “inserts itself into the client computer [and] take[s] steps to remedy the vulnerability.” *Id.*, [0016],

[0027] (“[t]he remedial exploit may rely on the same vulnerability that is the intended target of the original malicious exploit”). Remedies include “**removing software.**” *Id.*, [0028]. Removing a software component alters the computer system to “ameliorate or eliminate the malicious exploit’s ability to cause harm.” *Id.*, [0028], [0047].

332. **In sum**, the Oberheide-Patsos-Roytman-Wolff combination teaches “wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system [Wolff remedying the vulnerability that is the intended target of the exploit by removing software].”

a. Motivation to Combine Oberheide with Wolff

333. It would have been obvious, and a POSITA would have been motivated to modify Oberheide’s remediation techniques in light of Wolff’s teaching to remove the software to prevent future exploitation.

334. A POSITA would have been motivated to add an option to remove a vulnerable software component, as taught by Wolff (EX1009), in scenarios where a simple patch will not fully resolve the issue. Mell (EX1044), 2-11 (“Removing or uninstalling the affected software or vulnerable service eliminates the vulnerability and any associated threat.”). For example, Oberheide’s preferred actions to “resolve or reduce vulnerabilities” include “downloading, accessing, and/or selecting a patch

or software update that can be applied to the system.” Oberheide (EX1005), [0026], [0013] (teaching employing either “a persistent fix [or] a non-persistent computing system patch”).

335. Merely reducing a vulnerability or applying a temporary patch, however, may not fully resolve the issue. Mell (EX1044), 2-8 (pointing out that some software configurations may be intended as “temporary remediation measure[s]” until a persistent patch is released). As such, Wolff’s modification enhances Oberheide’s options to facilitate remediation by including additional techniques, such as removing the exploited software, to more completely resolve a vulnerability.

336. A POSITA would have appreciated a reasonable expectation of success making this modification. Both Oberheide and Wolff provide methods of exploiting vulnerabilities for self-healing purposes. Oberheide (EX1005), Abstract, [0013], [0019], Fig. 6; Wolff (EX1009), [0014]-[0016], [0027]-[0028]. As such, modifying Oberheide’s method of self-healing to include the ability to remove the vulnerable software component would have required simple programming techniques known to a POSITA (*e.g.*, invoking existing OS tools used to uninstall a software package). This modification would have used the known technique of software component removal to improve Oberheide’s system to yield the predictable result of removing the exploited vulnerability. Mell (EX1044), 2-11 (“[R]emoving unnecessary

software and services[] and running only what is essential for the system's purpose is a recommended security practice.”).

337. **In sum**, Oberheide's system provides actions for mitigating vulnerabilities, such as applying a patch or software update. As modified by Wolff (EX1009), Oberheide's system may additionally act by removing a vulnerable software component (*i.e.*, “removing a software component of the computer system that is utilized by the set of exploits to attack the computer system”), thereby completely eliminating the vulnerability and its associated threat.

3. Claim 13: “The method of claim 8, wherein altering the configuration of the computer system comprises removing a software component of the computer system that is utilized by the set of exploits to attack the computer system.”

338. Oberheide (EX1005), as further modified by Wolff (EX1009), teaches altering the configuration of the computer system by removing a software component of the computer system that is utilized by the set of exploits to attack the computer system. *See* §IX.A.2 above, Claim 7.

X. CONCLUSION

339. 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: August 15, 2025

By: 

Erez Zadok, PhD