IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

KEYSIGHT TECHNOLOGIES, INC.,
Petitioner,

v.

CENTRIPETAL NETWORKS, INC.,
Patent Owner.

U.S. Patent No. 10,567,343
Issue Date: February 18, 2020

Inter Partes Review No.: Unassigned

DECLARATION OF DR. STUART STANIFORD IN SUPPORT OF PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 10,567,343

I. Introduction 1

II. List of Documents I Considered in Forming My Opinions 3

III. My Background and Qualifications 15

IV. Person of Ordinary Skill in the Art (POSA) 19

V. Relevant Legal Standards 21

VI. The '343 Patent 24

A. Description 24

B. Priority Date of Claims 36

C. Antecedents to the '343 Patent 36

D. Prosecution History of the '343 Patent 39

E. Prior Attempted IPR of the '343 Patent 42

VII. Technical Background and State of the Art 48

A. Computer Network Architectures and Communication Protocols 49

B. HTTP Protocol 50

C. Network Security Systems – Firewalls 56

D. Network Security Systems – Intrusion Detection Systems and Intrusion Prevention Systems 60

E. Detecting Exfiltration of Data 64

F. Inspecting and Applying Packet Filtering Rules to Encrypted and Unencrypted SSL/TLS Record Protocol Data 71

VIII. Claim Construction 78

A. "Operator" 79

IX. Challenge 1: Sourcefire and the Emerging Threats Rules Render Claims 1-20 Obvious 88

A. Introduction 88

B. The Sourcefire Reference (EX-1004) 97

C. The '343 "Operator" as expressed in Sourcefire's Snort Rules 108

D. The Emerging Threats Rules 123

E. Detailed Invalidity Analysis 141

1. Independent Claims 1, 8, and 15 141

2. Dependent Claims 2, 9, and 16 161

3. Dependent Claims 3, 10, and 17 162

4. Dependent Claims 4, 11, and 18 164

5. Dependent Claims 5 and 12 166

6. Dependent Claims 6, 13, and 19 169

7. Dependent Claims 7, 14, and 20 172

X. Objective Indicia of Non-Obviousness 176

XI. Conclusion and Declaration 177

I. Introduction

1. I am over the age of eighteen (18) and otherwise competent to make this declaration.

2. I have been retained by Keysight Technologies, Inc. ("Keysight" or "Petitioner") as an independent expert consultant in this proceeding before the United States Patent and Trademark Office. I am being compensated for my time in connection with this matter at my standard consulting rate of $475 per hour. My compensation in no way depends on the outcome of this Inter Partes Review, and I have no other interest in this proceeding.

3. This petition for *Inter Partes* Review involves U.S. Patent No. 10,567,343 ("the '343 Patent") (EX-1001). The '343 Patent is titled "Filtering Network Data Transfers," lists Sean Moore as the inventor, and identifies Centripetal Networks, Inc. ("Centripetal" or "Patent Owner") as the assignee. The '343 Patent issued February 18, 2020, from U.S. Patent Application No. 15/614,956 that was filed June 6, 2017, and is a continuation of and claims priority to a series of patent applications including U.S. Patent Application No. 14/625,486 filed on February 18, 2015, now U.S. Patent No. 9,686,193 ("the '193 patent"), and U.S. Patent Application No. 13/795,822 file on March 12, 2013, now U.S. Patent No. 9,124,552 ("the '552 patent"). EX-1001; EX-1002; EX-1063; EX-1043.

4. For the purposes of this Inter Partes Review, as I discuss later, I have been instructed to assume that the effective filing date of the Claims of the '343 Patent challenged by the Petitioner in this *Inter Partes* Review is the earliest possible priority date, i.e., the March 12, 2013 filing date of U.S. Patent Application No. 13/795,822.

5. The '343 Patent is directed to systems and methods for filtering data packets in a network that match possible threat rules in order to detect and prevent exfiltrations of sensitive data. EX-1001, 1:62-2:9, 2:54-58. I am familiar with the technology described in the '343 Patent as of the earliest possible priority date of March 12, 2013.

6. In preparing this Declaration, I have reviewed the '343 Patent (EX-1001), the file history of the '343 Patent (EX-1002), and each of the documents cited below. I have considered these documents in light of the general knowledge in the art as of March 12, 2013. In formulating my opinions, I have relied upon my experience in the relevant art. I have also considered the viewpoint of a person of ordinary skill in the art ("POSA") of network security systems, including firewalls, intrusion prevention systems, data-leak prevention systems, and other packet filtering devices for protecting computer networks from cyber-attacks, as of March 2013.

7. I have been asked to provide my technical expertise, analysis, insights and opinions regarding the '343 Patent and relevant references that form the basis of the grounds of rejection set forth in the accompanying Petition for *Inter Partes* Review of the '343 Patent. As described in detail below, I offer the following opinions in this Declaration:

- A POSA would have found Claims 1-20 of the '343 Patent to be obvious over the Sourcefire 3D System User Guide Version 4.10 (EX-1004 - "Sourcefire") in light of published Emerging Threats Snort rules intended for use with the Sourcefire system and its underlying Snort detection engine (EX-1020 - "Emerging Threats").

Emerging Threats has not previously been considered by the Board or the Patent Office during examination of the application that issued as the '343 patent, in a prior petition for *Inter Partes* Review of the '343 patent, or in relation to examination or post-institution challenges of related patents. Emerging Threats will serve to document the range of things the intrusion detection community used Snort and Sourcefire to detect (and allow or block) prior to the priority date of the '343 patent, including numerous techniques for detecting exfiltrations.

II. List of Documents I Considered in Forming My Opinions

8. In formulating my opinions, I have considered and relied on statements in the documents identified below. These documents include patents,

patent applications, learned treatises, periodicals, pamphlets, technical standards documents, textbooks, and other publications. I consider each of the references below as a reliable authority for the statements on which I rely.

Exhibit No.	Title
EX-1001	U.S. Patent No. 10,567,343 ("the '343 Patent")
EX-1002	File History of the '343 Patent ("File History")
EX-1004	Sourcefire 3D System User Guide Version 4.10 ("Sourcefire"), *available* at https://www.scribd.com/document/96860221/Sourcefire-3D-System-User-Guide-v4-10
EX-1005	IPR2018-01760 CS-1005: Declaration of John Leone
EX-1006	Bard, G.V., "Cryptology ePrint Archive: Report 2006/136: A Challenging but Feasible Blockwise-Adaptive Chosen-Plaintext Attack on SSL," archived at Internet Archive Wayback Machine on May 3, 2006 at https://web.archive.org/web/20060503043443/http://eprint.iacr.org:80/2006/136
EX-1007	Barry M. Leiner et al., "A Brief History of the Internet," Internet Society, 1997, available at: https://www.internetsociety.org/wp-content/uploads/2017/09/ISOC-History-of-the-Internet_1997.pdf
EX-1008	U.S. Pat. No. 7,032,031 ("Jungck")

Exhibit No.	Title
EX-1009	E. H. Spafford, "The Internet Worm Program: An Analysis," *Purdue Technical Report CSD-TR-823*, 1988, available at: https://spaf.cerias.purdue.edu/tech-reps/823.pdf
EX-1010	Eichin, M.W. and Rochlis, J.A., "With Microscope and Tweezers: The Worm from MIT's Perspective," *Communications of the ACM*, Volume 32 Number 6, June 1989; *available at* https://www.cs.jhu.edu/~fabian/courses/CS600.624/Rochlis_morris.pdf
EX-1011	T. Eisenberg *et al.*, "The Cornell Commission: On Morris and the Worm," *Communications of the ACM*, Volume 32 Number 6, June 1989; *available* at https://www.cs.cornell.edu/courses/cs1110/2009sp/assignments/a1/p706-eisenberg.pdf
EX-1012	J. C. Mogul, "Simple and Flexible Datagram Access Controls for Unix-based Gateways," Digital Western Research Laboratory Technical Report No. 89/4, 1989; *available* at https://www.hpl.hp.com/techreports/Compaq-DEC/WRL-89-4.pdf
EX-1013	*Intentionally Left Blank*
EX-1014	*Intentionally Left Blank*
EX-1015	*Intentionally Left Blank*
EX-1016	Heberlein, L. T.L, *et al.*, "A Network Security Monitor," *Proceedings, 1990 IEEE Computer Society Symposium on Security and Privacy*, 1990, *available at* https://ieeexplore.ieee.org/document/63859.

Exhibit No.	Title
EX-1017	IPR2018-01760 CS-1017: "NetRanger Network Security Management System User Guide". v1.3.1 *Wheelgroup Corporation*, 1997.
EX-1018	Roesch, M., "Snort: Lightweight Intrusion Detection for Networks," *Lisa* (Vol. 99, No. 1, pp. 229-238), 1999, *available* at https://www.usenix.org/legacy/event/lisa99/full_papers/roesch/roesch.pdf
EX-1019	U.S. Pat. No. 8,370,936 ("Zuk")
EX-1020	Emerging Threats, available at https://web.archive.org/web/20101202025325/http://rules.emergingthreats.net/open/snort-2.8.4/emerging-all.rules (Dec. 2, 2010).
EX-1021	Authorized Classification and Control Markings Register, Vol. 1, Ed. 2 (Version 1.2), May 12, 2008, available at http://sgp.fas.org/othergov/intel/register.pdf
EX-1022	"Transmission Control Protocol," *IETF RFC 793*. J. Postel, ed., September 1981, archived at Internet Archive Wayback Machine on February 2, 2007 at https://web.archive.org/web/20070202201546/https://tools.ietf.org/html/rfc793.
EX-1023	"Internet Protocol," *IETF RFC 791*, J. Postel, ed., September 1981, archived at Internet Archive Wayback Machine on February 4, 2007 at https://web.archive.org/web/20070204151303/http://tools.ietf.org/html/rfc791.
EX-1024	"File Transfer Protocol," *IETF RFC 765*, J. Postel, ed., June 1980, archived at Internet Archive Wayback Machine on

Exhibit No.	Title
	February 6, 2007 at https://web.archive.org/web/20070206005843/https://tools.ietf.org/html/rfc765.
EX-1025	T. B. Lee, "The Original HTTP as defined in 1991," World Wide Web Consortium, 1991, archived at Internet Archive Wayback Machine on June 5, 1997 at https://web.archive.org/web/19970605071155/https://www.w3.org/Protocols/HTTP/AsImplemented.html.
EX-1026	I. Ristic, "SSL/TLS and PKI History," April 2018, available at https://www.feistyduck.com/ssl-tls-and-pki-history/.
EX-1027	Service Name and Transport Protocol Port Number Registry, Internet Assigned Numbers Authority (IANA), available at https://www.iana.org/assignments/service-names-port-numbers/service-names-port-numbers.xhtml
EX-1028	Understanding /etc/password File Format, available at https://www.cyberciti.biz/faq/understanding-etcpasswd-file-format/
EX-1029	ASCII Table, available at https://www.asciitable.com/
EX-1030	Affidavit of Nathaniel E Frank-White, Internet Archive, Archive.org (Dec. 12, 2022)
EX-1031	*Intentionally Left Blank*
EX-1032	K. Ingham & S. Forrest, "A History and Survey of Network Firewalls," *University of New Mexico Computer Science Department Technical Report 2002-37*, 2002, *available* at https://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf.

Exhibit No.	Title
EX-1033	Sourcefire Support Website Webpage, archived at Internet Archive Wayback Machine on April 30, 2011, *available* at https://web.archive.org/web/20110430194135/https:/support.sourcefire.com/ ("Sourcefire Support Webpage").
EX-1034	Crow, J., Bennet, B.; Structure of Social Security Numbers, available at https://www.eff.org/pages/structure-social-security-numbers
EX-1035	*Curriculum Vitae* of Stuart Staniford
EX-1036	Webster's New World Computer Dictionary, Tenth Edition (2003)
EX-1037	Microsoft Computer Dictionary, Fifth Edition (2002)
EX-1038	T. Dierks & E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.1," IETF RFC 4346, April 2006, archived at Internet Archive Wayback Machine on June 26, 2006 at https://web.archive.org/web/20060626134135if_ /http://www.ietf.org:80/rfc/rfc4346.txt
EX-1039	*Intentionally Left Blank*
EX-1040	Minoli, Daniel, et al., Minoli-Cordovana's Authoritative Computer & Network Security Dictionary (2006)
EX-1041	Dictionary Of Computing & Communications, McGraw-Hill (2003)
EX-1042	IPR2018-01436 CS-1003: Declaration of Dr. Stuart Staniford in Support of Petition for Inter Partes Review of US Patent No. 9,124,552.

Exhibit No.	Title
EX-1043	U.S. Patent No. 9,124,552
EX-1044	IPR2018-01437 CS-1003: Declaration of Dr. Stuart Staniford in Support of Petition for Inter Partes Review of US Patent No. 9,160,713
EX-1045	IPR2018-01760 CS-1038, Declaration of Christopher Butler
EX-1046	IPR2018-01760 CS-1042, Declaration of Jacob H. Baugher III
EX-1047	U.S. Patent No. 8,356,357
EX-1048	U.S. Patent No. 8,141,127
EX-1050	M. Edwards, "Bleeding Edge of Snort," Oct. 10, 2004, available at https://www.itprotoday.com/windows-78/bleeding-edge-snort
EX-1051	Bleedingsnort.com, Wayback Machine, Internet Archive, available at https://web.archive.org/web/*/http://www.bleedingsnort.com/
EX-1052	*Intentionally Left Blank*
EX-1053	Emerging Threats Rule Documentation Wiki, available at https://doc.emergingthreats.net/
EX-1054	Proofpoint Emerging Threats Rules, available at https://rules.emergingthreats.net/
EX-1055	*Intentionally Left Blank*
EX-1056	*Intentionally Left Blank*

Exhibit No.	Title
EX-1057	*Intentionally Left Blank*
EX-1058	U.S. Patent No. 9,160,713
EX-1059	*Intentionally Left Blank*
EX-1060	*Intentionally Left Blank*
EX-1061	*Intentionally Left Blank*
EX-1062	IPR2018-01559 CS-1003: Declaration of Dr. Stuart Staniford in Support of Petition for Inter Partes Review of US Patent No. 9,686,193
EX-1063	U.S. Patent No. 9,686,193
EX-1064	*Intentionally Left Blank*
EX-1065	*Intentionally Left Blank*
EX-1066	*Intentionally Left Blank*
EX-1067	Cisco Systems, Inc., v. Centripetal Networks, Inc., IPR2018-01436 at Paper 40 (Final Written Decision) (U.S. Pat. 9,124,552)
EX-1068	Cisco Systems, Inc., v. Centripetal Networks, Inc., IPR2018-01437 at Paper 40 (Final Written Decision) (U.S. Pat. 9,160,713)
EX-1069	Centripetal Networks, Inc. v. Cisco Sys., 2020-1635 (Fed. Cir. Mar. 10, 2021), available via https://cafc.uscourts.gov/opinions-orders/20-1635.opinion.3-

Exhibit No.	Title
	10-2021_1745854.pdf (decision on appeal from '552 and '713 FWDs)
EX-1070	Cisco Systems, Inc., v. Centripetal Networks, Inc., IPR2018-01559 at Paper 7 (Decision Denying Institution of Inter Partes Review) (U.S. Pat. 9,686,193)
EX-1071	*Intentionally Left Blank*
EX-1072	*Intentionally Left Blank*
EX-1073	"Hypertext Transfer Protocol -- HTTP/1.1" IETF RFC 2616. R. Fielding, et al., June 1999, archived at Internet Archive Wayback Machine on January 9, 2005 at https://web.archive.org/web/20050119021312/https://www.ietf.org/rfc/rfc2616.txt.
EX-1074	HTTP-Requests, available at https://www.tutorialspoint.com/http/http_requests.htm
EX-1075	HTTP – Responses, available at https://www.tutorialspoint.com/http/http_responses.htm
EX-1076	IPR2021-01155 CS-1003: Declaration of Dr. Robert Akl in Support of Petition for Inter Partes Review of US Patent No. 10,567,343
EX-1077	Palo Alto Networks, Inc., v. Centripetal Networks, Inc., IPR2021-01155 at Paper 12 (Decision Denying Institution of Inter Partes Review) (U.S. Pat. 10,567,343)
EX-1078	Emerging Threats FAQ, available at https://doc.emergingthreats.net/bin/view/Main/EmergingFAQ

Exhibit No.	Title
EX-1079	*Intentionally Left Blank*
EX-1080	Snort rules 2.4 ftp.rules, available via https://web.archive.org/web/20060112054053/http://www.snort.org/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.4.tar.gz
EX-1081	Declaration of Dr Sean Moore in Support of Centripetal's Responses to Non-Final Office Action in Re-examination of '193 patent.
EX-1082	Centripetal Networks Press Release, "Centripetal Networks Inc. Adds Emerging Threats IQRisk® Rep List and IQRisk Query for Their RuleGate® Appliance". Downloaded from https://www.centripetal.ai/press-releases/centripetal-emerging-threats-rulegate/
EX-1083	Cisco Systems, Inc., v. Centripetal Networks, Inc., IPR2018-01559, Patent Owner's Preliminary Response. (U.S. Pat. 9,686,193).
EX-1084	Palo Alto Networks, Inc., v. Centripetal Networks, Inc., IPR2021-01155, Patent Owner's Preliminary Response. (U.S. Pat. 10,567,343).
EX-1085	Ex-Parte Re-examination History for Patent 9,686,193.
EX-1086	Palo Alto Networks, Inc., v. Centripetal Networks, Inc., IPR2021-01155, Paper 16: Decision Denying Petitioner's Request for Rehearing.
EX-1087	Cisco Systems, Inc., v. Centripetal Networks, Inc., IPR2018-01436, Patent Owner's Preliminary Response. (U.S. Pat. 9,124,552).

Exhibit No.	Title
EX-1088	Gartner, August 2011. Magic Quadrant for Content-Aware Data Loss Prevention. Obtained by SS from https://www.etruserve.com.tw/gtb/GTB%20wp/rp-gartner-magic-quadrant-dlp-2011.pdf
EX-1089	US Patent No. 8,565,108.
EX-1090	US Patent No. 7,467,202.
EX-1091	*Intentionally Left Blank*
EX-1092	*Intentionally Left Blank*
EX-1093	*Intentionally Left Blank*
EX-1094	S. Turner & C. Polk, "Prohibiting Secure Sockets Layer (SSL) Version 2.0," IETF RFC 6176, March 2011, archived at Internet Archive Wayback Machine on March 18, 2011 at https://web.archive.org/web/20110318184421/https://tools.ietf.org/html/rfc6176.
EX-1095	M. Chatel, "Classical vs. Transparent IP Proxies," *IETF RFC 1919*, March 1996.
EX-1096	E. Rescorla, "HTTP over TLS", *IETF RFC 2818*, May 2000, archived at Internet Archive Wayback Machine on January 25, 2007 at https://web.archive.org/web/20070125003236/https://tools.ietf.org/html/rfc2818.
EX-1097	J. Moser, "The First Few Milliseconds of an HTTPS Connection", *Moserware*, June 10, 2009, archived at Internet Archive Wayback Machine on June 13, 2009 at

Exhibit No.	Title
	https://web.archive.org/web/20090613041022/http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html
EX-1098	Apple Quicktime RTSP URL buffer overflow. IBM advisory for CVE-2007-0015, 2007, available at https://exchange.xforce.ibmcloud.com/vulnerabilities/31203
EX-1099	Real Time Streaming Protocol (RTSP). RFC 2326, April 1998. Available at https://www.rfc-editor.org/rfc/rfc2326.html

9. The documents I cite to in this declaration are the types of documents that experts in my field ordinarily rely on to understand the state of the art and what was known or obvious as network security technology evolved over time. The documents I cite to also comport with my knowledge and experience working in the field. For the exhibits that include a hyperlink, I personally downloaded the documents and confirmed that the exhibits are true copies of the documents that are, or were, available from those links. Many of the exhibits listed above were also exhibits in IPR2018-01436, IPR2018-01437, and IPR2018-01559 for which I also provided declarations. I reviewed the exhibits for this IPR and recognize the overlapping exhibits as being the same documents that were marked as exhibits in IPR2018-01436 and the others. For example, I recognize EX-1004 in this IPR as the same Sourcefire user guide identified as CS-1004 in IPR2018-01436.

10. A different petitioner, Palo Alto Networks, also filed an IPR against the '343 patent, and provided a declaration by Dr Robert Akl. EX-1076. Although in significant part I analyze different prior art and consider different arguments than Dr. Akl, in a small number of places I adopted reasoning and text of his that I agreed with into this declaration. I have also borrowed from my own prior declarations on the same patent family where the text and reasoning are applicable to the '343 patent. EX-1042, EX-1044, EX-1062.

III. My Background and Qualifications

11. My expertise qualifies me to do the type of analysis required in this case in the field of network security systems, including firewalls, network intrusion prevention systems, and other packet filtering devices for protecting computer networks from cyber-attacks.

12. Throughout the remainder of this Declaration, I will refer to the field of network security systems, including firewalls and other packet filtering devices for protecting computer networks from cyber-attacks, as the "relevant field" or the "relevant art." In formulating my opinions, I have relied upon my training, knowledge, and experience in the relevant art. A copy of my current curriculum vitae is provided (EX-1035), and it provides a comprehensive description of my background, experience, and academic and employment history, including a list of

network security publications for the last 28 years, and a list of cases in which I have testified as an expert witness at trial or by deposition.

13. In 1988, I received a Bachelor of Science degree in Mathematical Physics from the University of Sussex, UK. In 1990, I received a Master of Science degree in Physics from the University of California at Davis ("UC Davis"), and then in 1993 a PhD in Physics from the same institution. My PhD thesis focused on building and analyzing computer simulations of statistical physics systems. In 1995, I received a Master of Science in Computer Science from UC Davis, having transferred into the network security group there. My Master's thesis focused on a technique for tracing computer attackers back to their true location on the Internet.

14. From 1994 to 1999, I worked at UC Davis as a researcher and then an Assistant Adjunct Professor. During this time, I led a team of 10 researchers and graduate students building a distributed network intrusion detection system, GrIDS. GrIDS was the first multi-layer hierarchical intrusion detection system and focused on correlating data across large networks to detect worms and other distributed attacks. I was then asked by the Defense Advanced Research Projects Agency ("DARPA") to lead a working group of over a hundred researchers and

developers building intrusion detection systems to create a Common Intrusion Detection Framework ("CIDF") for those systems to interoperate.

15. From 1998 to 2004, I founded and ran my own company, Silicon Defense. I managed a staff of up to 23 people, performing a mixture of government contract research and commercial product development. I obtained ten research contracts for the company of up to $2.3M in size, working for four different DARPA program managers. I also chaired an IETF working group developing an intrusion detection exchange format, and served on a number of conference program committees. I wrote a series of technical research papers with colleagues: two concerned analysis and improvement for the popular open source Snort network intrusion detection system, which is directly relevant to the subject matter of this declaration. After five years of 100%+ annual growth out of cashflow, the company was obliged to file bankruptcy due to DARPA's decision to classify further network security research.

16. From 2004 to 2005, I was Principal Scientist at Nevis Networks. Here I architected a very high-speed intrusion event correlation system and also the traffic anomaly detection subsystem for a 10 Gigabit per second ("Gbps") network security switch.

17. From 2005 to 2007, I was a solo consultant doing business as Invicta Consulting. I performed technical consulting for startups – finding security vulnerabilities and false negatives in products, designing algorithms, and analyzing network security products for effectiveness. I also worked as litigation consultant and expert witness for several law firms – I analyzed patents and prior art, and in the case of SRI vs ISS, I wrote an expert report, defended my analysis at deposition, and testified at trial.

18. From 2008 to 2013, I was Chief Scientist at FireEye (having had them as a consulting client since 2006). I designed and implemented novel statistical algorithms to detect exploit websites installing malware on client computers. This allowed FireEye to repurpose its existing product line, which was previously failing in the marketplace, as a solution to the drive-by download (malicious exploit website) problem. The company successfully raised two rounds of venture capital based on this new value proposition, the new product was very successful, and eventually FireEye went public in September 2013. The share price went from 7 cents when I joined to a peak over $95 as of early 2014, when it was worth $10 billion. I also extended the system I built to detect attacks buried inside PDF, Java and Flash formats. I was a member of the management team throughout my tenure

at FireEye, reporting directly to the founder and CEO, and participated in all executive planning and operational discussions.

19. From 2013 to 2016 I was an Adjunct Professor at Cornell University. I taught a yearly graduate course on applied network security, which included several lectures covering Snort and the Snort rule format.

20. From 2014 to 2016, I was founder and CEO of Cayuga Networks, a startup developing a high-speed network web application attack detection system.

21. Since 2017, I have been President of Staniford Systems, LLC. I consult for a variety of companies on technical matters, advise startups in the network security field, and provide expert consulting for litigation matters.

22. I have co-authored a number of publications in the network security area, which have attracted over 11,000 citations in the academic literature. I am a named inventor on twelve U.S. Patents: Nos 8,850,571, 8,990,939, 8,997,219, 9,106,694, 9,118,715, 9,438,622, 9,519,782, 9,954,890, 10,027,690, 10,282,548, 10,432,650, and 10,757,120.

IV. Person of Ordinary Skill in the Art (POSA)

23. I am familiar with the knowledge and capabilities of one of ordinary skill in the art. A POSA in March 2013 would have had a working knowledge of packet-switched networking, firewalls, security policies, communication protocols and layers, user interfaces, and the use of customized rules to filter cyber-attacks in

the network. A POSA would have had a bachelor's degree in computer science, computer engineering, or an equivalent, and four years of industry experience. Lack of work experience can be remedied by additional education, and vice versa.

24. I have been informed and understand that a POSA is a hypothetical person who is presumed to be aware of all pertinent prior art, thinks along conventional wisdom in the art, and is a person of ordinary creativity.

25. With respect to the '343 Patent, in my opinion, a POSA in March 2013 would have been familiar with firewalls and other packet filtering devices for protecting computer networks from cyber-attacks, and have a working knowledge of packet-switched networking, firewalls, security policies, communication protocols and layers, and the use of customized rules to address cyber-attacks. A POSA would have gained knowledge of these concepts through a mixture of training and work experience. The work experience and education levels may vary between persons of ordinary skill, with some persons having a Bachelor's degree in computer science, computer engineering, or the like and four years of industry experience, others holding an advanced degree in the above disciplines, but having fewer years of experience, and others having no formal education but experience in computer programming and network security of at least eight years.

26. A POSA may work as part of a multi-disciplinary team and draw upon not only his or her own skills, but also take advantage of certain specialized skills of others on the team to solve a given problem.

27. In the course of my career in network security I have employed and supervised numerous junior engineers at a variety of commercial product companies and thus have a good sense of the typical knowledge and capability of POSAs in the network security field. I have also taught network security courses at the university level, including teaching a course titled "Defending Computer Networks" to Cornell Masters' students in 2013.

V. Relevant Legal Standards

28. I am not a lawyer and will not provide any legal opinions. Although I am not a lawyer, I have been informed and understand that certain legal standards are to be applied by technical experts in forming opinions regarding the meaning and validity of patent claims. I have been asked to provide my opinions regarding whether the claims of the '343 Patent would have been obvious to a person having ordinary skill in the art at the time of the alleged invention, in light of the prior art.

29. I have been informed and understand that a patent claim is not patentable under 35 U.S.C. § 103 if the differences between the patent claim and the prior art are such that the claimed subject matter as a whole would have been obvious at the time the claimed invention was made to a person having ordinary

skill in the relevant art. Obviousness, as I have been informed and understand, is based on the scope and content of the prior art, the differences between the prior art and the claim, the level of ordinary skill in the art, and, to the extent that they exist, certain objective indicia of non-obviousness.

30. I understand that objective indicia (or "secondary considerations") can be important evidence regarding whether a patent is obvious or nonobvious, if it has an appropriate nexus to the claimed invention, i.e., is a result of the merits of a claimed invention (rather than the result of design needs or market-pressure advertising or similar activities). Such indicia include: commercial success of products covered by the patent claims; a long-felt need for the invention; failed attempts by others to make the invention; copying of the invention by others in the field; unexpected results achieved by the invention as compared to the closest prior art; praise of the invention by the infringer or others in the field; the taking of licenses under the patent by others; expressions of surprise by experts and those skilled in the art at the making of the invention; and the patentee proceeded contrary to the accepted wisdom of the prior art.

31. I have been informed that whether there are any relevant differences between the prior art and the claimed invention is to be analyzed from the view of a person of ordinary skill in the relevant art at the time of the invention. As such,

my opinions below as to a person of ordinary skill in the art are as of the time of the invention, even if not expressly stated as such; for example, even if stated in the present tense.

32. In analyzing the relevance of the differences between the claimed invention and the prior art, I have been informed that I must consider the impact, if any, of such differences on the obviousness or non-obviousness of the invention as a whole, not merely some portion of it. The person of ordinary skill faced with a problem is able to apply his or her experience and ability to solve the problem and also look to any available prior art to help solve the problem.

33. I have been informed that a precise teaching in the prior art directed to the subject matter of the claimed invention is not needed. I have been informed that one may take into account the inferences and creative steps that a person of ordinary skill in the art would have employed in reviewing the prior art at the time of the invention. For example, if the claimed invention combined elements known in the prior art and the combination yielded results that were predictable to a person of ordinary skill in the art at the time of the invention, then this evidence would make it more likely that the claim was obvious. On the other hand, if the combination of known elements yielded unexpected or unpredictable results, or if the prior art teaches away from combining the known elements, then this evidence

would make it more likely that the claim that successfully combined those elements was not obvious.

34. I have been informed and understand that there are recognized, exemplary, rationales for combining or modifying references to show obviousness of claimed subject matter. Some of the rationales include the following: combining prior art elements according to known methods to yield predictable results; simple substitution of one known element for another to yield predictable results; use of a known technique to improve a similar device (method or product) in the same way; applying a known technique to a known device (method or product) ready for improvement to yield predictable results; choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; 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; and some 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 prior art teachings to arrive at the claimed invention.

VI. The '343 Patent

A. Description

35. The ’343 Patent is directed to systems and methods for filtering data packets leaving a network to ensure that the data packets are not improperly transferring sensitive data or credentials from the network, a category of cyber-attacks known as exfiltrations. For example, exfiltrations are often facilitated by using popular network data transfer protocols, such as the Hypertext Transfer Protocol (HTTP) used by the World Wide Web. Such attacks may appear to an insufficiently careful observer as normal network behavior; older network trust models, such as those used by firewalls that do not interpret application layer data like HTTP, may interpret exfiltrations as trusted operations. EX-1001 1:28-51.

36. The ’343 Patent describes a two-stage filtering process for stopping such attacks implemented at a Packet Security Gateway (PSG) examining packets

that are exiting a trusted network towards an external network. EX-1001 8:39-52.

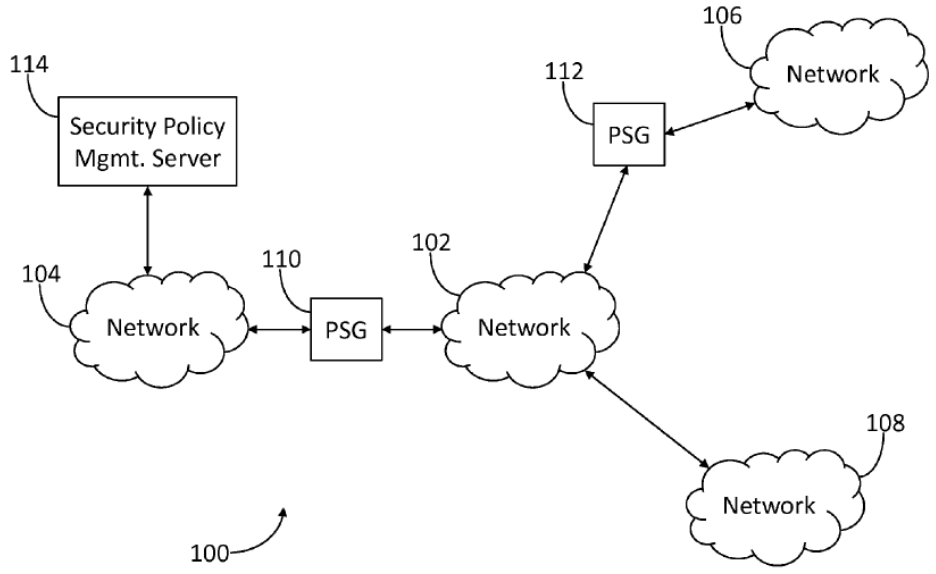


FIG. 1

37. FIG. 1 of the ’343 Patent (above) illustrates a network environment 100 of various networks protected by PSGs 110 and 112. Networks 104, 106, and 108 may be private networks such as Local Area Networks (LANs) or Wide-Area Networks (WANs) operated by various organizations (*e.g.*, one or more commercial enterprises, companies, universities, military commands, government agencies, or cyber-criminal organizations). Network 102 represents the public Internet functioning as an interconnect between networks 104, 106 and 108. EX-1001 3:31-45. Networks 104 and 106 are protected by PSGs 110 and 112, which in turn are administered by security policy management server 114. A PSG (110 or 112) receives packets and applies one or more filters to allow or block

transmission of the received packets according to rules specified in the security policy management server 114. EX-1001 4:15-27; 3:62-4:14; 5:29-49.

38. As illustrated in FIG. 1, networks 104 and 106 may each form part of a single organization's protected or secured network associated with security policy management server 114. For example, PSG 110 may protect network 104 from one or more cyber-attacks (*e.g.*, exfiltrations) mediated by network 102 (*e.g.*, the Internet), and PSG 112 may protect network 106 from one or more cyber-attacks (*e.g.*, exfiltrations) mediated by network 102. EX-1001 4:27-41.

39. In operation, each PSG performs a two-stage filtering process for received packets. In a first stage, the "5-tuple" of packet header fields are filtered (comprising source and destination Internet Protocol (IP) addresses, transport protocol, and source and destination ports as defined in either the Transmission Control Protocol (TCP), or User Datagram Protocol (UDP)). In a second stage, application packet header field values are filtered. Conceptually, the first stage may determine if the network security policy allows any communications between the resources identified in the 5-tuple rule; if so, the second stage may determine if the policy allows the specific method or type of communication (e.g., file read, file write, encrypted communication, etc.) between the resources. EX-1001 8:39-52.

40. FIG. 4 of the ’343 Patent (below) illustrates the two-stage filtering process performed at a PSG:

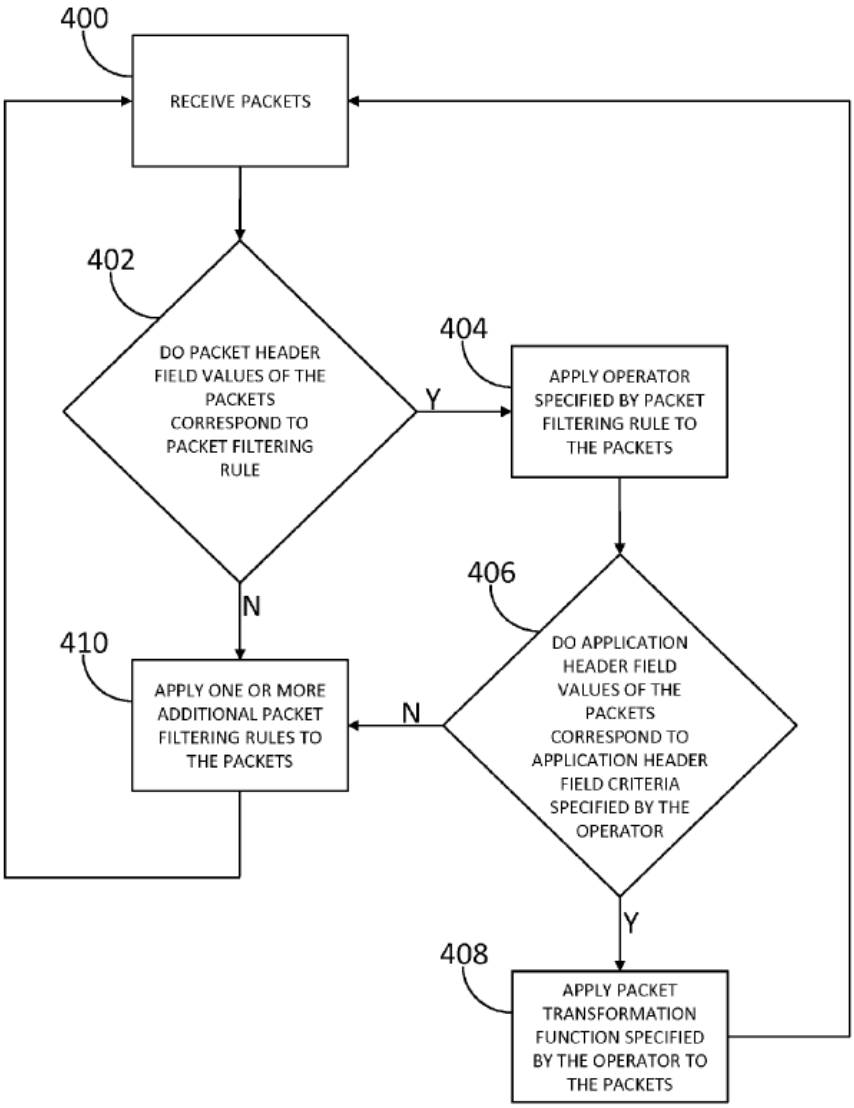


FIG. 4

41. At step 400, packets are received, *e.g.*, as they are exiting network 104 while destined for network 106. At step 402, a determination is made as to

whether some received packets have packet header field values corresponding to a packet-filtering rule. For example, a determination may be made as to whether the 5-tuple header values (*i.e.*, transport protocol, source IP address, source port, destination IP address, or destination port) correspond to a rule provided by the security policy. EX-1001 9:18-30.

42. At step 404, responsive to determining that the received packets have packet header field values corresponding to the packet-filtering rule, an operator specified by the rule may be applied to the received packets. EX-1001 9:30-34. FIG. 3 of the ’343 Patent (below) identifies example operators that may be specified by a rule for packet header field values.

43. At step 406, a determination is made as to whether an application header field value of the received packets corresponds to a corresponding criteria specified by the operator. EX-1001 9:37-42.

44. At step 408, having determined that the received packets have an application header field value that so corresponds, a packet transformation function specified by the operator is applied to the received packets. Notwithstanding the name “packet transformation function”, the ’343 Patent describes that this function can include encapsulating the packets into a tunnel, altering one or more header

field values, as well as allowing or blocking transmission of the packets. EX-1001 7:41-47*;* 9:47-61.

45. In one embodiment, the ’343 Patent describes that an operator can include (1) determine if an IP packet traversing a boundary contains an HTTP packet (*e.g.*, an application-level HTTP packet) that specifies one or more specific HTTP methods (*e.g.*, GET, PUT, POST, etc.), and (2) allow the packet *(e.g.*, if a GET method is specified), or block the packet (*e.g.*, if a PUT or POST method is specified). EX-1001 2:41-47; FIG. 3 (below) (rule 6).

46. FIG. 3 of the ’343 Patent illustrates the rules and operators that may be specified by the security policy for the specific example of packets sent from

network 104 and received at PSG 110.

218

Five-tuple

Rule #	IP Protocol	Source IP Address	Source Port	Destination IP Address	Destination Port	Operator
1 (302)	TCP	140.210.*	*	140.212.*	22	ALLOW
2 (304)	TCP	140.210.*	*	140.212.*	25	ALLOW
3 (306)	TCP	140.210.*	*	140.212.*	110	ALLOW
4 (308)	TCP	140.210.*	*	140.212.*	143	ALLOW
5 (310)	TCP	140.210.*	*	140.212.*	443	REQUIRE-TLS-1.1-1.2
6 (312)	TCP	140.210.*	*	214.*	80	HTTP-EXFIL
7 (314)	*	*	*	*	*	BLOCK

FIG. 3

47. FIG. 3 illustrates seven example rules, each of which may specify criteria and one or more operators that may be applied to packets matching the specified criteria. In Stage I, the specified criteria are based on the 5-tuple of values in the packet header information for a received packet. EX-1001 5:50-64.

48. For example, rule 1 may specify that IP packets containing one or more TCP packets, originating from a source IP address that begins with 140.210 (network 104), having any source port, destined for an IP address that begins with 140.212 (network 106), and destined for port 22 (*e.g.*, associated with the Secure

Shell (SSH) protocol used for system logins) should have an ALLOW operator (which means the PSG should transmit the packets). Similarly, rule 2 instructs the PSG to ALLOW similar packets destined for port 25 (*e.g.*, associated with the Simple Mail Transfer Protocol (SMTP) for e-mail transfers to a mail server). Likewise, rule 3 instructs the PSG to ALLOW similar packets destined for port 110 (*e.g.*, associated with Post Office Protocol version 3 (POP3), used to download e-mail into an email client). Similarly, rule 4 instructs the PSG to ALLOW packets destined for port 143 (*e.g.*, associated with the Internet Message Access Protocol (IMAP) used to obtain e-mail from a server by an email client). EX-1001 5:65-6:24.

49. However, Rule 5 instructs the PSG that packets with similar packet header information except being destined for port 443 (*e.g.*, associated with the port for the Hypertext Transfer Protocol Secure (HTTPS) protocol) should have a specified Transport Layer Security (TLS) protocol operator applied to them in a second stage. EX-1001 6:25-33. Rule 5 allows web browsers attached to network 104 to conduct HTTPS sessions (*e.g.*, secure web sessions) with any web servers attached to network 106, but requires the field value in the headers of application data contained in IP packets to specify version 1.1 or 1.2 of the Transport Layer Security (TLS) protocol "because the popular TLS version 1.0 protocol has a

known security vulnerability that attackers may exploit to decrypt HTTPS sessions." EX-1001 6:60-7:3, 7:41-47, 8:11-38. In the specific embodiment disclosed for Rule 5, the '343 Patent states:

> For an HTTPS session composed of IP packets, the application packets contained in the IP packets may be TLS Record Protocol packets. The header fields of TLS Record Protocol packets may not be encrypted. One of the header fields may contain a value indicating the TLS version.

EX-1001 8:14-19.

50. Rule 6 instructs the PSG that packets with similar packet header information except being destined for port 80 (*e.g.*, associated with the HTTP protocol) on network host addresses beginning with 214.* (*i.e.* associated with network 108) should have an HTTP-EXFIL operator applied to them in a second stage[1]. An HTTP-EXFIL operator may allow HTTP packets containing a GET method, but may block HTTP packets containing other HTTP methods (*e.g.*, PUT, POST, CONNECT, etc.). Such an operator allows a web browser to "surf the

[1] Figure 3 of the '343 patent has rule 6 as being for packets bound to 214.*, but the description in the text specification refers to 200.214.* at 7:7, but 214.0.0.0/8 at 7:26. My assumption is this is an error and 200.214* and 214.* are intended to be the same IP address range: *i.e*. that associated with network 108 in Fig 1.

web" (*e.g.*, download web pages hosted by web servers using the GET method), but may prevent the web browser from writing files to a web server (using the PUT method), posting forms (*e.g.*, forms that might contain login credentials, such as usernames or passwords) to a web server (using the POST method), or otherwise communicating with a web server (using any HTTP method except GET). The '343 Patent describes that, "[b]ecause attackers may often use HTTP PUT or POST methods to exfiltrate sensitive data, operators such as HTTP-EXFIL of rule 6 may be used to stop such exfiltrations." EX-1001 7:4-23.

51. Rule 7 is a "wildcard" rule and applies a BLOCK operator to any packets that do not match the criteria of any of Rules 1-6. In other words, Rules 1-7 are applied to packets successively in descending order. EX-1001 6:34-39.

52. While rules 1-4 identify operators that may be satisfied based on first stage review of packet header information only, rules 5-6 identify operators that may also require a second stage review of the application header information. One function of operators like HTTP-EXFIL (rule 6) and REQUIRE-TLS-1.1-1.2 (rule 5) may be to inspect fields in the headers of application data contained in IP packets, determine field values, and depending on the field values, decide to allow or block the packets, encapsulate the packets into a tunnel, or alter one or more header field values in the packets, etc. EX-1001 7:41-47, 9:47-61.

53. By enforcing sets of rules similar to rules 1-7, the ’343 Patent states that companies may allow their employees to freely surf the web and conduct company business, but may block one or more cyber-attacks (*e.g.*, HTTP-mediated exfiltrations). EX-1001 7:35-40.

54. The claims of the ’343 Patent are directed to a method, apparatus and computer-readable media for implementing the two-stage packet filtering process at the PSG to block exfiltrations. Independent Claim 1 is directed to the method; independent Claim 8 is a corresponding apparatus claim performing the steps recited in Claim 1; and independent Claim 15 is a corresponding computer–readable media programmed to perform the steps recited in Claim 1.

55. Each of the two-stage packet-filtering embodiments of the ’343 Patent discloses the following combination to purportedly enable increased network security: (1) determining whether 5-tuple packet header field values of a received packet correspond to packet header criteria of a rule, (2) determining whether application header field values of the received packet correspond to application header criteria of the rule, and (3) allowing or blocking the received packet based on these determinations and the operator specified in rule. EX-1001 6:25-33, 7:4-23, 7:41-8:38, 8:39-52; FIG. 3 (rules 5, 6). The ‘343 Patent describes that this same combination was used to allow or block received packets using rules with

various types of application header criteria including: (a) particular HTTP methods (EX-1001 *7:4-8:7*, FIG. 3 (rule 6)), (b) particular encryption protocol (TLS) versions (EX-1001 6:25-33, 8:7-38, FIG. 3 (rule 5)), and other application header criteria (*e.g.*, particular data transfer protocols (*e.g.*, FTP, RTP), particular messaging protocols (*e.g.*, XMPP, etc.) (EX-1001 8:53-9:10).

B. Priority Date of Claims

56. The patent application that issued as the '343 Patent was filed on June 6, 2017, as a continuation application that claimed priority through a series of patent applications, the earliest of which is U.S. Patent Application No. 13/795,822, now the '552 Patent, filed on March 12, 2013.

57. For the purposes of the *Inter Partes* Review, I have been instructed to assume that the effective filing date of each of the challenged claims is March 12, 2013.

C. Antecedents to the '343 Patent

58. As described above, the '343 Patent claims priority to the '552 patent, which issued on Sep. 1, 2015. The specifications of the '552 and '343 patents are identical, but the claims of the '552 patent are different, focused on blocking or allowing HTTPS packets based on the value of the TLS-version value in the packets. EX-1043, 11:5-35.

59. The Board instituted an IPR of the '552, in IPR2018-01436 (for which I provided a declaration) and held a trial. All twenty-one claims of the '552 patent were declared obvious and thus unpatentable over Sourcefire (specifically use of the ssl_version keyword in a rule). EX-1067 pp. 70-71. Patent Owner appealed the decision to the Court of Appeals for the Federal Circuit, which affirmed the correctness of the Board's decision to declare all claims obvious. EX-1069 pp. 17-21. The decision also affirmed that all challenged claims of U.S. Patent No. 9,160,713 (EX-1058), another patent claiming priority to the '552 patent, were invalid as obvious in view of Sourcefire. EX-1069 at 17-21; EX-1068.

60. After the issue of the '552 patent, Patent Owner obtained patent 9,686,193 (the '193 patent), a continuation of the '552 patent which issued on June 20, 2017. The specification was identical to the '552 patent (and the '343 patent). However, the claims differed from the '552 patent in that they didn't focus on any particular protocol, but rather the general idea of a rule specifying a pair of operators to block certain kinds of data transfers through to a particular destination network, while allowing other kinds of data transfer. EX-1063, 11:28-62.

61. The '193 patent was challenged in IPR2018-01559 by petitioner Cisco Systems, Inc ("Cisco") who argued that the '193 patent, like its '552 forbear, was obvious over Sourcefire (I also provided a declaration for this IPR). EX-1062.

However, unlike in the '552 case, the Board declined to institute an IPR of the '193 patent, stating that the petition and my declaration failed to make clear what the term "operator" meant in the patent, and as a result failed to demonstrate a reasonable likelihood of prevailing in the IPR. EX-1070 pp. 8-13.

62. Subsequently Cisco requested an *Ex Parte* Reexamination of claims 18 and 19 of the '193 patent. EX-1085 pp. 231-326. In this request, Cisco argued that the arguments and determinations of the Board in the '552 patent IPR were sufficient to render the '193 claims also unpatentable. EX-1085 pp. 261-267. Initially the examiners were persuaded by these arguments and issued a Non-Final Office Action declaring that the previously approved claims were now denied as being obvious over Sourcefire. EX-1085 pp. 158-179.

63. However, Patent Owner then argued that the "operator" as claimed in the '193 patent was distinct from the operator in the '552 patent claims in that the operator was not part of the rules, but separately called by the rules and could be called by multiple rules. EX-1085 pp. 34-37, 125-130. Also, they argued that the purpose of the '193 patent was to block exfiltrations, and that Sourcefire does not provide any motivation to a POSA to write rules targeting exfiltrations since it was advertised as an intrusion prevention system, and the Sourcefire manual does not mention the word "exfiltration". EX-1085 pp. 37-40, 130-133. These contentions

were supported by their expert, Prof. Orso, a computer science professor with a specialty in software engineering. EX-1085 pp. 32-45. I have reviewed the re-examination history (EX-1085) and can find no sign that either Patent Owner or Prof. Orso disclosed to the Patent Office that there were many publicly available Snort rules, compatible with Sourcefire, that targeted exfiltrations (see Section IX D, below).

64. The Patent Office accepted Patent Owner's arguments and issued a re-examination certificate sustaining the '193 claims. EX-1085 pp. 5-7.

D. Prosecution History of the '343 Patent

65. I reviewed the file history of the '343 Patent (EX-1002).

66. On September 27, 2018, the Examiner issued an Office Action rejecting all twenty of the pending claims. Claims 1-3, 6, 8, 13, and 15 were rejected as unpatentable on the ground of non-statutory double-patenting over claims 1-3, 16, 18, and 19 of the '193 Patent. EX-1002 pp. 281-282. Claims 1-3, 5, 8-10, 12, and 15-17 were rejected under pre-AIA 35 U.S.C. § 103(a) as unpatentable over U.S. Pub. No. 2011/00559916 ("Ahn") in view of U.S. Pub. No. 2003/0005122 ("Freimuth"). EX-1002 pp. 290-299. The Examiner added other references for rejections of the other dependent claims. EX-1002 pp. 299-305.

67. On March 27, 2019, the Applicant responded by amending the claims, including amendments to independent claims 1, 8, and 15 to differentiate between

"a first criterion" specified to examine a packet header and a "second criterion" specified to examine an "application header field value," to recite a "matching" packet-filtering rule, and "an prevent exfiltration operation" (*sic*) among other changes. EX-1002 pp. 271-278. The Applicant argued that Ahn fails to teach or suggest a determination based on an application header field value. The Applicant further argued that Freimuth fails to "teach or suggest processing packets comprising data corresponding to a first criterion specified by at least one matching packet-filtering rule" because after the TCP handshake completes and data is received to determine the HTTP header, the kernel of Freimuth classifies the request based on the header values. *Id*., 268-269.

68. On June 3, 2019, the Examiner again rejected all twenty pending claims. The Examiner rejected claims 1, 8, and 15 for failing to comply with the written description requirement of 35 U.S.C. § 112 (pre-AIA), first paragraph because the specification did not convey that the "second portion of packets is a portion of the first portion of packets." EX-1002 pp. 96-98. The Examiner also rejected the claims under pre- AIA 35 U.S.C. § 103(a). Claims 1-3, 5, 8-10, 12, and 15-17 were rejected as unpatentable over Ahn in view of U.S. Pub. 2012/0084866 ("Stolfo"). EX-1002 pp. 99-101. The Examiner added other references for

rejections of the dependent claims. EX-1002 pp. 101-115. The Examiner objected to the ungrammatical "an prevent exfiltration operation". EX-1002 p. 98.

69. On November 4, 2019, the Applicant filed a Request for Continued Examination and amended independent claims 1, 8, and 15 to recite that at least one packet transformation function configured to "prevent an exfiltration operation, where the at least one packet transformation function indicates whether each packet in the second portion of packets is allowed to continue toward its destination". The revised claim also removed the limitation that the second portion of packets is a portion of the first portion of packets (which the examiner had argued was not disclosed by the specification). EX-1002 pp. 77, 79-82. In response to the Examiner's rejections, Applicant argued that the references do not disclose "apply[ing], to each packet in the second portion of packets that match the second criterion, at least one packet transformation function configured to prevent an exfiltration operation, wherein the at least one packet transformation function indicates whether each packet in the second portion of packets is allowed to continue toward its destination" as recited in independent claims 1, 8, and 15. EX-1002 pp. 84-88. In particular, Applicant argued that Stolfo instead disclosed generating decoy traffic to confuse or detect attackers, rather than applying a

transformation to existing network traffic, and thus does not prevent an exfiltration operation. EX-1002 p. 87-88.

70. A Notice of Allowance issued on December 12, 2019, in which the Examiner pointed to the Applicant's November 4, 2019 arguments with respect to independent claims 1, 8, and 15, and the amendments to claim 1, as the reason for allowance. EX-1020 pp. 40-41. Thus, the '343 Patent claims seem to have been allowed because the Examiner concluded that the prior art of record failed to disclose a two-stage packet filtering process limited to preventing exfiltration operations. EX-1086 pp. 7-8.

E. Prior Attempted IPR of the '343 Patent

71. In IPR2021-01155, another petitioner, Palo Alto Networks, Inc ("Palo Alto"), challenged the validity of the '343 patent, and provided a supporting declaration signed by Dr Robert Akl, a computer engineering professor with a research background mainly in techniques to improve wireless networking. EX-1076. The Board declined to institute the IPR. EX-1077. In this section, I briefly review the arguments raised by Dr. Akl and Palo Alto, and the Board's reasons for declining to proceed.

72. The Akl declaration reviewed the prosecution history (much as I did above in Section VI C), and concluded its review stating:

> Thus, the ’343 Patent claims were apparently allowed because the Examiner concluded that the prior art of record failed to disclose a two-stage packet filtering process.

EX-1076 p. 61. Akl did not in that paragraph note the Examiner’s consideration of the exfiltration term in the independent claims.

73. For most of the Claims (except for 7, 14, and 20), the Akl declaration presented a detailed invalidity analysis based on Sourcefire alone, which was done in the form of claim charts. I highlight the argument the Akl declaration makes for the ‘343 patent Claim 1 element that the operator is “configured to prevent an exfiltration operation”, which can be summarized with the following quotes:

> To the extent Sourcefire does not explicitly describe a rule in which an HTTP method such as GET, PUT, or CONNECT is specifically included as a keyword, a person of ordinary skill in the art would have found obvious a rule using the HTTP Method as part of the criteria to invoke the HTTP inspect preprocessor to identify traffic using that HTTP method (*e.g.*, PUT), and either pass or drop those packets based on the inspection.

EX-1076 p. 88. Also:

> Additionally, a person of ordinary skill in the art would have understood that HTTP methods such as PUT, POST, and CONNECT were well-known methods of network exfiltration, because a malicious actor can use such methods to send data out of a network.

EX-1076 p. 89. Thus, the Akl declaration did not identify an explicit teaching in Sourcefire of an exfiltration method, but argued it was well-known in the art to use HTTP methods for this purpose.

74. The Akl declaration also relies on a patent application "Mantripragada" for the real-time protocol related claims of the '343 (Claims 7, 14, and 20). EX-1076 p. 71.

75. It's noteworthy that the Akl declaration didn't discuss or cite to any discussion of the '193 IPR 2018-01559, or the Board's declining to institute the same, or the *Ex Parte* re-examination of the '193 patent. EX-1076 pp. 15-18. Thus there was a lack of understanding or discussion of factors the Board and the Patent Office had found relevant in deciding to uphold the '193 patent during various prior proceedings.

76. In response to the Palo Alto Networks petition, and Dr Akl's declaration in support of it, Patent Owner filed a preliminary response (EX-1084). Patent Owner did not provide any expert declaration in support. EX-1084 pp. i-ii. Patent Owner argued that the Board should deny institution based on the following factors:

a. Based on the state of the co-pending litigation. EX-1084 pp. 4-9.

b. Because of petitioner's alleged violation of word count limits. EX-1084 pp. 10-13.

c. Because the Patent Office had extensively considered Sourcefire previously and the petition lacked new arguments. EX-1084 pp. 25-38.

d. Because the '343 patent (and the '193 patent before it) are sufficiently distinguished over the prior art as not to be obvious. EX-1084 pp. 40-49.

e. The petition's alleged hindsight bias. EX-1084 pp. 49-50.

f. Because the petition allegedly didn't sufficiently consider secondary considerations of obviousness. EX-1084 pp. 50-51.

77. In its decision to deny institution, the Board did not address points a), b), e), and f). EX-1077 pp. 13-14. Accordingly, I now only summarize Patent Owner's arguments in c) and d) that the '343 claims have limitations different enough from the prior art, and that Sourcefire had been extensively considered by the Board.

78. Patent Owner argued that the Patent Office had been aware of Sourcefire in its prosecution of the '343 patent, in that it was listed on the face of the patent, and disclosed in an IDS signed by the Examiner. EX-1084 p. 27. The

Patent Office also was in receipt of disclosures of the petitions and expert declarations in IPR2018-01436 (of the grandparent '552 patent) and for IPR2018-01559 (of the parent '193 patent). EX-1084 p. 28. Further, the Patent Owner argued that the Patent Office had considered the parent '193 patent in relation to Sourcefire in the re-examination proceeding over that patent, and that the petition and Akl declaration ignore this fact. EX-1084 p. 35. Specifically, they assert, the '343 patent claim 1 is directed to preventing exfiltrations, and in the re-examination, this element was found lacking in Sourcefire. EX-1084 p. 31, 35-36.

79. Patent Owner also argued directly at length that the '343 patent was not obvious over Sourcefire, specifically because it did not disclose a technique for preventing exfiltrations and that a POSA would not be motivated to write such rules. EX-1084 pp. 40-49. For example:

> Here, Petitioner's obviousness ground attempting to fill the exfiltration gap is legally insufficient as it completely fails to allege a motivation to modify *Sourcefire* to prevent exfiltration or a reasonable expectation of … success in doing so.

EX-1084 p. 48.

80. In its decision, the Board largely accepted Patent Owner's arguments, and found that the Examiner had considered Sourcefire in the '343 prosecution process, and essentially similar arguments to those that Palo Alto raised in its

petition had already been considered. EX-1077 pp. 14-19. In particular, the Board focused on the exfiltration element of the claims:

> Petitioner has not shown that the two-stage filtering process was pertinent to the Examiner's allowance of the claims. Consequently, Petitioner has not identified sufficiently any material error on the part of the Examiner during prosecution of the '343 patent.

EX-1077 p. 17. Also:

> And, although there are similarities between the '552 patent's claims and those of the '343 patent, there are also material differences that Petitioner does not address. For example, the '343 patent's claims recite determination steps and applying the packet transformation function configured to prevent an exfiltration operation. These features are not present in the '552 patent's claims, notwithstanding Petitioner's argument that the '343 patent's claims are broader than the '552 patent's claims. Hence, we are not persuaded that the Examiner erred regarding the '552 patent or filings from the *inter partes* review in which it was challenged.

EX-1077 p. 18 (internal citations removed).

81. In the remainder of this Declaration, I will document that using intrusion prevention systems to block exfiltrations in fact had a long history before the priority date of the '343 patent, and that Snort rules to block exfiltration were well known, including some rules that were extremely similar in their operation to the approaches described in the '343 patent.

VII. Technical Background and State of the Art

82. This section describes the historical development of the art for network security systems, including firewalls and other packet filtering devices for protecting computer networks from cyber-attacks, leading up to the state of the art as of March 2013. The prior art references, and the discussions of what was known to a POSA, provide the factual support for the general description of the state of the art at the time before the priority date of the '343 Patent. This section provides additional motivation to combine the teachings of the Sourcefire reference (EX-1004) and the associated Emerging Threats ruleset with the knowledge, skill, and creativity of a POSA, justifies why a POSA would have had a reasonable expectation of success in doing so, rebuts any assertion by the Patent Owner of unpredictability in the art, and rebuts any assertion by the Patent Owner of unexpected results.

83. As of March 2013, a POSA working in the field of network security systems, including firewalls and other packet filtering devices for protecting computer networks from cyber-attacks, would have been familiar with certain common concepts and related components including, but not limited to: computer network architectures and communication protocols, firewalls, Intrusion Detection Systems and Intrusion Prevention Systems, the layering of computer network

protocols, the use of customized rules to prevent cyber-attacks, and applying such rules to unencrypted and encrypted packet data.

A. Computer Network Architectures and Communication Protocols

84. The Internet has its roots in conceptual papers on packet switching in the 1960s. These papers led to the funding by DARPA of an experimental internetwork – the ARPANET – with the first two nodes being connected in late 1969, and then growing steadily through the 1970s as a means of doing research on computer networking. EX-1007. The early ARPANET used a protocol, NCP, that is no longer in use and proved not to be scalable.

85. NCP's deficiencies led to the development of two new protocols – Internet Protocol (IP) to provide best effort delivery of data packets across heterogeneous networks (EX-1023), and Transmission Control Protocol (TCP) to provide reliable delivery of streams of data across an IP internetwork (EX-1022). These two protocols were described in the IETF Request For Comment (RFC) documents RFC 791 and 793 respectively in 1981. EX-1023; EX-1022. The ARPANET cut over to using the new TCP/IP protocol suite on January 1, 1983. That date can be viewed as the birth date of the modern Internet. EX-1007.

86. Logically, the Internet is a web of intermediate network nodes and communication paths interconnecting those network nodes that provide multiple data transmission routes from any given point to any other given point on the

network (*i.e.*, between any two computers.) EX-1008 3:59-4:11. The disparate networks are connected where they join by special devices – IP routers – that decide which way to send each packet. A router typically creates or maintains a table of the available routes and their conditions and uses this information to determine the best route for a given packet. EX-1008 3:32-46, 38:29-39:8.

87. While TCP/IP provided the basic abstract facility to move data across a global network, higher-level application protocols were required to allow data communication to be managed conveniently. Application protocol data would be transmitted in the body of packets with IP and TCP headers. Particularly important applications included email (with protocols such as SMTP, POP and IMAP), with early email systems dating back to 1972. EX-1007. Other important protocols included FTP for making files available, first defined in 1980 (EX-1024), and the Hypertext Transfer Protocol (HTTP) for facilitating the exchange of hypertext markup language (HTML) and World Wide Web (WWW) content, first defined in 1991 (EX-1025). We now turn to the HTTP protocol, which figures prominently in the '343 patent.

B. HTTP Protocol

88. Due to its relevance to many of the examples in this IPR declaration, and in particular to the exfiltration examples in the '343 patent, I briefly sketch here the operation and syntax of the HTTP protocol. HTTP was first defined by

Tim Berners Lee at the CERN Physics Lab in Geneva, Switzerland in 1991. EX-1025. I will particularly focus on version 1.1, which was published in 1999, and which was still the version in wide currency in March 2013. EX-1073.

89. HTTP runs in the simplest case over a single TCP connection between an application on one host – the HTTP client – which initiates the connection, and a second application (likely on a different host) – the HTTP server – which answers the connection. The protocol is a request/response oriented one, in which the client (for example a web browser) makes a series of requests in a special text format over the connection, and the HTTP server responds with a set of responses to each request in the order to which they were received over the other side of the connection. EX-1073 p. 12[2].

90. A simple example of a request will serve to make the discussion more concrete:

```
POST /cgi-bin/process.cgi HTTP/1.1
User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)
Host: www.tutorialspoint.com
Content-Type: application/x-www-form-urlencoded
Content-Length: length
```

[2] Throughout this declaration, when citing to IETF RFC documents, I use the internal page numbers of the original RFC, not exhibit page numbers where they differ.

```
Accept-Language: en-us
Accept-Encoding: gzip, deflate
Connection: Keep-Alive

licenseID=string&content=string&/paramsXML=string
```

EX-1074. The request message is a series of ASCII text lines terminated with a two-byte sequence "\r\n" (CRLF - ASCII characters 13 and 10). EX-1073 p. 31, EX-1029. As is conventional, the example request message above suppresses the two-byte sequence at the termination of each line for clarity.

91. The first line is the request line and consists of a one-word "method" such as 'GET', 'POST', or "CONNECT'. EX-1073 p. 36. 'GET' for example indicates that the request is intended to obtain some data from the server. EX-1073 p. 53. Meanwhile 'POST' (as in the example above) is intended for the use case of the HTTP client supplying data to the server (and possibly receiving something in response). EX-1073 p. 54. Following the method is a URI (a general case of a URL) indicating the particular resource on the server that the data is to be obtained from, or supplied to. EX-1073 pp. 36-38. In the example above, the request URI being posted to is '/cgi-bin/process.cgi' – the path to a script that the server will invoke to process this request. Finally, the request line ends with a version string, in this case 'HTTP/1.1'. EX-1073 pp. 17-18, 36.

92. Following the request line are a series of request header lines which can appear in any order, and which are of the form "Fieldname: field-value\r\n". EX-1073 pp. 31-32, 38-39. In the above example, we see for instance "User-Agent: Mozilla/4.0 (compatible; MSIE5.01; Windows NT)". The User-Agent can be used to identify the software producing the request (in this case a version of Microsoft's Internet Explorer). EX-1073 p. 145. We also see "Connection: Keep-Alive" which is a request from the browser to the server to keep the underlying TCP (or other) transport connection open for further requests after servicing this one.

93. The HTTP request header concludes with a blank line (that is an additional CRLF beyond that required to terminate the last header line). Then follows an optional message body. In the example above, this is "licenseID=string&content=string&/paramsXML=string" which appears to be the submission to the server of the results of filling out a form on a web page.

94. Each HTTP request from the client will, in the absence of a network or software fault, result in an HTTP response from the server. HTTP responses follow a similar design philosophy to HTTP requests. An example response is as follows:

```
HTTP/1.1 200 OK
Date: Mon, 27 Jul 2009 12:28:53 GMT
```

```
Server: Apache/2.2.14 (Win32)
Last-Modified: Wed, 22 Jul 2009 19:15:56 GMT
Content-Length: 88
Content-Type: text/html
Connection: Closed

<html>
<body>
<h1>Hello, World!</h1>
</body>
</html>
```

EX-1075. Like the request, the response message header is a series of ASCII text lines terminated with a two-byte sequence "\r\n" (CRLF - ASCII characters 13 and 10). EX-1073 p. 31, EX-1029.

95. The first line of the response provides the version of the protocol that the server is speaking (in this case HTTP/1.1). EX-1073 pp 17-18, 39-41. This is followed by a numerical status code, and a text exposition of the status code. In the example above, the code is "200 OK" indicating all is well with the request and the server's response to it. EX-1073 pp 40-41, 58-59. A variety of other codes indicate various conditions and problems (e.g. "404 NOT FOUND", various 500 series internal server errors, and so on). EX-1073 pp 40-41.

96. Subsequent lines of the response provide additional modifying information about the response or the entity attached to the response in the HTTP body. Like the request lines, they are in the form "Fieldname: field-value\r\n".

EX-1073 pp. 31-32, 41-42. In the example above, a "Content-Type" indicates that an HTML text (*i.e.* a web page) is attached to the response. Similarly, a "Content-Length" field indicates that the attached HTML text is 88 bytes long.

97. It's important to understand that there is no guaranteed relationship between the HTTP requests and responses and the underlying TCP/IP packets. The TCP code on the client will take the stream of request data from the web client, and break it into TCP segments. EX-1022 pp. 3-4. In principal, those could contain as little as a single byte of application data, in addition to the TCP header. EX-1022 p. 12. Furthermore, the IP software is at liberty to fragment TCP segments into small pieces – units of 8 bytes, which is smaller than a single TCP header – so that there can be packets on the wire that are part of the TCP stream but actually contain zero bytes of the actual HTTP layer data. EX-1023 pp. 8-9. Likewise, with small HTTP requests, it is entirely possible for multiple requests to end up in a single packet on the network, or for a packet to contain all or part of one request, and also part of another.

98. With normal non-malicious TCP/IP code, the most common situation is that a request which will fit in a single 1500 byte ethernet frame will in fact be all in one TCP segment inside one IP packet inside one ethernet frame. However the nature of exfiltration detection (the intended point of the '343 patent) is that

often both the HTTP client inside the protected network and the HTTP server outside the protected network are under the control of someone with malicious intent (*i.e.* the intent to leak data in a way that violates network policy). If so, the client is free to packetize data in any manner that will result in its successful delivery but minimize the chance of detection by security systems in the middle of the transmission path.

99. In particular, during my time working with FireEye between 2007 and 2013 I implemented and maintained a system that reconstructed HTTP requests and responses from packets seen at an IPS system targeting the detection of malicious web servers. Therefore, I spent extensive time examining attack traffic between such servers and compromised clients on our customer's networks. It was not at all uncommon to see HTTP requests and responses split carefully over multiple packets in such a way as to place potentially suspicious content divided between two packets.

100. This background on HTTP should serve to make the later discussions of exfiltration detection in the '343 patent, and via Snort signatures in the Emerging Threats ruleset, more comprehensible.

C. Network Security Systems – Firewalls

101. Throughout the mid to late 1980s, the Internet grew very rapidly as more and more researchers and developers connected departmental mainframes,

minicomputers, Unix workstations, and the like to it. The Internet reflected the openness and information sharing that characterizes the research community that developed it, with few security barriers beyond the requirement to login on remote systems (which often had weak or default passwords). EX-1007.

102. This era of complete openness spawned a variety of abuses by computer hackers. A particularly spectacular incident sparked the development of the modern network security industry: the Internet worm of 1988. In this episode, a Cornell graduate student, Robert Morris Jr., released a worm – a self-replicating program capable of infecting other computers across the network via vulnerabilities in their operating system or applications. This worm infected about 6,000 computers across the Internet. Most of these computers were disabled, either by large numbers of copies of the worm clogging their memory, or at least by the need to disinfect and upgrade them. Many of the computers were important shared facilities with numerous users, so much of the work on the Internet was disrupted. The incident caused a great deal of analysis and soul-searching amongst the Internet community. Within a year (by 1989), both the first firewall and the first network intrusion detection system (NIDS) were under development. EX-1009, EX-1010, EX-1011.

103. The development of firewalls had its roots in the late 1980s practice of using filtering rules on Internet routers. This development began primarily as a way to prevent misconfigurations on one network that generated a lot of noise packets from affecting an adjacent network. However, it then resulted in the development of security applications, with routers providing simple facilities to block packets based on their source and destination networks and TCP ports. For example, with Cisco routers in 1989: "one could deny access to all but one host on a particular network. The Cisco gateways also allow discrimination based on IP protocol type, and TCP or UDP port numbers." EX-1012 p. 3. This example illustrates that the basic functionality of blocking or admitting packets based on a rule was available prior to 1989.

104. Early packet filters reasoned packet-by-packet without keeping any state information about prior packets. It quickly became apparent that this strategy was both expensive (in that possibly large firewall rulesets had to be separately processed for every packet, even in a long connection) and had security weaknesses (as hackers found tricky ways to fool packet filters into allowing probe packets through because the filter didn't know the state of the TCP connection). This led to the development of stateful inspection firewalls. Early stateful inspection firewalls applied the rules to the first packet in a TCP connection (the

"syn packet"), and then kept track in an in-memory hash table of allowed connections and their current state, meaning that subsequent packets in the connection could be allowed or blocked based on a single hash lookup. EX-1032; EX-1019 4:12-30.

105. By the mid-1990s, multiple systems had been developed which incorporated application functionality "transparently" into stateful inspection firewalls. EX-1095. The way this worked is that the firewall would parse the application data inside packets it was passing, and investigate whether they might be significant to application traffic routing. For example, the Zuk patent, filed in 2002, stated that:

> A packet filtering firewall with stateful inspection also has the ability to examine a packet in order to allow certain types of commands within an application while disallowing others. For example, a stateful inspection firewall can allow the FTP "get" command while disallowing the "put" command." EX-1009 4:31-41.

106. Based on understanding the application protocol, the firewall determined that certain TCP connections needed to be routed through a proxy. It would instantiate the relevant proxy code in its own memory and then connect the incoming client TCP connection to that proxy automatically, which would then perform authentication and provide proxy service on to the server. Thus, no later than 1996, a POSA familiar with then-current firewall technology would know that

some firewalls were discriminating whether to pass or block packets based on examining both the 5-tuple (*e.g.*, TCP/IP headers), but also the application headers in the packet data bodies.

D. Network Security Systems – Intrusion Detection Systems and Intrusion Prevention Systems

107. In an independent line of development from firewalls, the Internet worm and other less serious incidents of the time inspired researchers to begin exploring systems specifically designed to look for intrusions on networks by examining the data packets crossing the network. Initially, network intrusion detection was done by placing network interfaces on broadcast networks into "promiscuous" mode to inspect all the traffic, which could then be copied into a user-mode application where it could be inspected and reasoned about. The earliest such system was the Network Security Monitor (NSM) developed at UC Davis in 1991. EX-1016. This evolved into the US Air Force's ASIM system, and a large part of the development team on that effort split off and formed a company, WheelGroup, which launched the first commercial network intrusion detection system (IDS), NetRanger, in 1996.

108. NetRanger had a general facility to write rules that matched arbitrary strings in the application data inside packets, in addition to matching on IP and TCP packet header values. For example, a 1997 user guide for the NetRanger

system illustrated how to log every HTTP GET request passing by in packets on the monitored network:

The following example shows how to log every HTTP GET on a network. The context info provided with the string will show what URL was used in the GET.

Web	SIGID	Port	Direction	Occurrences	String Match
RecordOfStringName	8001	80	1	1	"GET[ ]+"

EX-1017 p. 189.

109. So, from the very beginning of the network intrusion detection industry, it was known to a POSA to look at application header data, and HTTP methods in particular, when examining packets transiting a network. *Id.*

110. Vulnerabilities with various HTTP methods (including exfiltration vulnerabilities) were also known to a POSA; indeed, the 1999 HTTP RFC categorized HTTP GET and HEAD methods as "safe" methods and "other methods, such as POST, PUT, and DELETE, in a special way, so that the user is made aware of the fact that a possibly unsafe action is being requested." EX-1073 pp. 44, 46-50, 130-134.

111. The first widely adopted open-source NIDS, Snort, was initially released in 1998, and was documented in a conference paper in 1999. EX-1018. Like other NIDS before it, Snort allowed the user to write rules for detecting

particular network packets based on both the 5-tuple information in the packet IP and TCP headers, as well as strings found in packet application data. For example:

```
alert tcp any any -> 10.1.1.0/24 80 (content: "/cgi-bin/phf"; msg: "PHF
probe!";)
```

112. This rule (see EX-1018 p. 5, Fig. 5) shows that alerting was to be done based on TCP protocol packets from any source IP address and port, and destination IP addresses in the range 10.1.1.0/24, and with destination port 80 (characteristic of HTTP traffic). This particular content keyword (/cgi-bin/phf) matches a uniform resource locator (URL) used in a popular attack of the time, but content keywords in general could be used to find and match any text in application data in packets transiting the network. The Snort NIDS provided two-stage rules like those in the '343 patent, because the rules evaluated packets based on 5-tuple information, and for packets that matched the 5-tuple information evaluated strings found in packet application data, very often application header field values.

113. The earliest NIDS were strictly involved in detecting attacks; these systems worked off a copy of the packets and could only alert or log suspected intrusions, not directly block them. However, it was quickly realized that customers had a need to take automated action in some cases. Already by 1997, WheelGroup's NetRanger system, for example, could be used to reconfigure a

packet-filtering router to block all packets from the source IP address used in a detected attack. EX-1017 p. 4-26.

114. The need to block attacks quickly, as well as the need to resolve various complex ambiguities in the meaning of packet traffic that computer intruders used to get around the early NIDS systems, meant that the industry quickly developed systems that were "inline" in the traffic flow, and which could choose to drop or pass any given packet. These systems were known as intrusion prevention systems (IPS), with multiple companies launching such products in 2002, including Intruvert (later acquired by McAfee), Tipping Point (3Com then HP), and OneSecure (later acquired by NetScreen then Juniper).

115. These inline intrusion prevention systems were capable of applying intrusion detection rules that looked both at 5-tuple header data and application data (e.g. content) in every packet and making pass/alert/block decisions on each such packet. For example, the Zuk patent, filed in 2002 for OneSecure, describes an IPS architecture in which packets were inspected, and the 5-tuple in the packet header was used to lookup the correct rules in a flow table (hash table of all TCP connections or UDP pseudo-connections currently being tracked by the system). The rules described in Zuk then allowed regular expression matching of any packet

content in the application layer, and packet application data content matches would result in triggering the rule and dropping the packet. EX-1019 col. 18, Fig. 11.

116. Similarly, the Jungck patent from 2006 is directed to a "packet interceptor/processor apparatus" that "applies one or more rules to . . . intercepted packets which execute one or more functions on a dynamically specified portion of the packet and take one or more actions with the packets." EX-1008 Abstract. Actions taken on packets include "releasing the packet unmodified, deleting the packet, modifying the packet, logging/storing information about the packet or forwarding the packet to an external device for subsequent processing." EX-1008 Abstract. Further, "the rules may be dynamically modified by the external devices." *Id*.

E. Detecting Exfiltration of Data

117. In this section, I briefly sketch the history of detecting exfiltration of data in network monitoring security systems, as distinct from detecting inbound attacks or reconnaissance. Exfiltration was a concern from very early on.

118. For example, in the introductory rationale to the 1990 paper "A Network Security Monitor" which documented the very first network intrusion detection system, the paper discussed the kind of threats that needed to be detected:

> Consequently, network attacks or intrusions such as eavesdropping on information meant for someone else, illegally accessing information

> remotely, breaking into computers remotely, inserting erroneous information into files, and flooding the network thereby reducing its effective channel capacity are not uncommon.

EX-1016 p 1. The first two items on the list are about attackers accessing confidential information from off the local network – i.e., they are exfiltration concerns. Likewise, in a discussion of the "Post Attack Phase", the paper raised the possibility of an attack resulting in "Dissemination of data stored on the system." EX 1016 p. 3. The paper implemented a statistical model of normal network traffic, that would have been capable of detecting at least some exfiltrations when they occurred via unusual traffic patterns, and the paper explicitly discussed the possibility of detecting an attacker who could "use the *rcp* service to copy the password file to another host" (an example of exfiltration).

119. The 1996 NetRanger 1.3.1 manual (referenced earlier) documented a number of detection signatures specifically targeted to detect exfiltration in progress. For example, the product could detect a DNS Zone Transfer Request. EX-1017 p. 4-64. This was a request to get a large number of records from the Domain Name System, which if successful would be useful in mapping the network. Similar, the product could detect requests for all DNS records. EX-1017 p. 4-64. The product was also capable of detecting CGI scripts leaking the password file via HTTP. EX-1017 p. 4-74. This is an exfiltration detection (cf the

'343 patent's concern for "posting of login credentials (usernames and passwords), etc." to remote nodes). EX-1001 2:56-58. Likewise, the NetRanger product had a detection for TFTP password file leaking. EX-1017 p. 4-75. And, as noted above, the product was documented to be able to log every HTTP GET on the network. EX-1017 p. 4-80 (it would be obvious to a POSA that it could also log PUT or POST if it could log GET, since they are simply known alternative HTTP methods).

120. Similarly, the Zuk patent (filed in Feb 2002) describes an early IPS system. EX-1019. Zuk devotes eight columns of the patent to the "background of the invention", and provides an excellent summary of the prior art as of 2002. Already, at that time, two-stage inspection filtering on particular application commands was known:

> A packet filtering firewall with stateful inspection also has the ability to examine a packet in order to allow certain types of commands within an application while disallowing others. For example, a stateful inspection firewall can allow the FTP "get" command while disallowing the "put" command.

EX-1019 p. 21. This already discloses a technique that the '343 patent leans on for its attempted exfiltration detection over a decade later:

> … in order to enforce, via the PSG, a Web usage policy that may, for example, allow users to surf (e.g., GET) to one or more web sites

> attached to the Internet, but may prevent such user(s) from one or more of writing (e.g., PUT) data files or posting (e.g., POST) forms to one or more websites. For example, administrator(s) may utilize such functionality to prevent one or more exfiltrations (e.g., file transfers containing sensitive information, posting of login credentials (usernames and passwords), etc,) to network nodes (e.g. web sites) that they may not trust.

EX-1001 2:49-58.

121. The core NIDS engine in Sourcefire, Snort, was begun as an open source project in the late 1990s. EX-1018. From the beginning, the rulesets were open-source and accepted community contributions. I personally used Snort rules extensively in the late 1990s and early 2000s and was familiar with downloading them from the Snort.org website managed by Sourcefire. Numerous rules from early on targeted exfiltration detection. For example, the following rule appeared no later than 2006 in the ftp.rules file for the Snort 2.4 ruleset:

```
alert tcp $EXTERNAL_NET any -> $HOME_NET 21
(msg:"FTP passwd retrieval attempt";
flow:to_server,established; content:"RETR"; nocase;
content:"passwd"; reference:arachnids,213;
classtype:suspicious-filename-detect; sid:356;
rev:5;)
```

EX-1080 SID:356. This rule concerns access by remote clients (`$EXTERNAL_NET any`), seeking to obtain from local FTP servers (`$HOME_NET 21)` copies of the passwd file, which would also be a case of

exfiltration. Further examples from the Emerging Threats rules will be considered in Section IX D below.

122. In the 2000s, the term "Data Loss Prevention" (or "DLP") came to be widely used for techniques for preventing confidential data leaving organizations against policy, and in particular the term "network data loss prevention" was the most common term for the kinds of techniques the '343 patent describes. The '343 patent shows no awareness of this literature or market.

123. For example, by 2011, market research firm Gartner was already producing a Magic Quadrant for "Content-Aware Data Loss Prevention" products that listed 13 vendors. EX-1088 p. 4. I now briefly illustrate some of the ideas that existed in this space.

124. One of these vendors was Fidelis, which was assigned the Savchuk patent 7,467,202, filed in Sep 2003. It disclosed analyzing network packets using "multi-dimensional content profiling and/or weighted keyword-in-context" with a view to detecting "the extrusion of data" and stopping "data transfers deemed malicious" based on the detailed analysis of the content of the packets. EX-1090 Abstract.

125. The Mustafa patent (8,141,127), filed in 2006, discusses that the known prior art at that time included searching in network traffic for regular

expressions of structured information (such as bank card numbers) and keywords associated with sensitive information. However, these techniques are limited to only detecting exact matches to the target information, and much sensitive information may not match pre-specified patterns. The patent goes on to propose a deep semantic analysis to identify the meaning of the content in network transmissions (and other places) and determine a Security Profile Score summarizing the chance of the data being sensitive and a possible exfiltration. EX-1048.

126. The Barile patent (8,356,357) assigned to Symantec and filed in July 2009 discussed in its background section:

> Given the immensity of the problems associated with loss of confidential data, it is essential that companies prevent these types of losses. Data loss prevention (DLP) products are one mechanism for curbing data loss. DLP products have a number of mechanisms for identifying confidential information (e.g., deep content analysis, including using dictionaries, keywords, or regular expressions, using partial document fingerprinting, etc.). Network DLP products or gateway based solutions generally run on a company's internet network connection, and analyze network traffic for transmissions of confidential data.

EX-1047 1:36-46. It went on to discuss that hackers were transforming or encrypting exfiltrated files to avoid DLP products, and proposed a technique for

detecting files with confidential information that had been transformed. This illustrates that it was well known already by 2009 to detect and respond to ex-filtrations by examining content of packets, and inventors were moving on to disclose complex schemes for dealing with countermeasures to DLP/exfiltration solutions.

127. For another example, consider the Marshall et al patent (8,565,108) filed in Sep 2010. EX-1089. It taught techniques for inspecting network transmissions looking for sensitive information being leaked, such as bank card data, and making a determination based on where in the network it was going, to take a variety of policy responses (such as blocking, allowing, quarantining, etc.):

> At block 1212, the portion of the network flow is analyzed to determine whether there is a match to the DLP policy. The method may determine whether there are matches to context criteria and/or content criteria of the DLP policy. As one possible example, a network flow that includes a bank card number and is incoming to a sales department may not be Suspicious (e.g., it may be associated with a customer's order for a product) and may not trigger a match to the DLP policy. However, a network flow that includes a bank card number and is outgoing from an accounting department could be Suspicious (e.g., it may represent potential disclosure of confidential customer information) and may trigger a match to the DLP policy.

EX-1089 24:22-35.

F. Inspecting and Applying Packet Filtering Rules to Encrypted and Unencrypted SSL/TLS Record Protocol Data

128. The early versions of HTTP transmitted their data unencrypted ("in the clear"), leaving them vulnerable to eavesdropping and the like. To address this vulnerability, in 1995, Netscape Corporation released an implementation of the Secure Sockets Layer (SSL) protocol, which inserted an encrypted channel running on top of TCP, and underneath HTTP. The first public version of this SSL protocol, version 2, had security issues[3], and was rapidly replaced with version 3 in 1996. The IETF then became involved in standardizing these encryption protocols, and released TLS 1.0, which was a very minor improvement on SSL v3 in 1999. To address well-known security vulnerabilities in SSL v2, SSL v3, and TLS 1.0, the IETF released TLS 1.1 in 2006, and then TLS 1.2 in 2008. *See, e.g.*, EX-1006 ("The attack in this paper is an application of the blockwise-adaptive chosen-plaintext attack paradigm, and is the only feasible attack to use this paradigm with a reasonable probability of success. The attack will work for all versions of SSL,

[3] By March of 2011, the security issues with SSL v2 were so well-known that the IETF issued RFC 6176 which "require[d] that when Transport Layer Security (TLS) clients and servers establish connections, they never negotiate the connection of Secure Sockets Layer (SSL) version 2.0." EX-1094.

and TLS version 1.0. This vulnerability and others are closed in TLS 1.1 (which is still in draft status) . . . It is hoped this paper will encourage the deprecation of SSL and speed the adoption of . . . TLS 1.1/1.2 when they are finially [*sic*] released."); EX-1006. In RFC 4346 (TLS 1.1), the IETF stated:

> This document is a revision of the TLS 1.0 [TLS1.0] protocol, and contains some small security improvements. The major changes are:
>
> - The implicit Initialization Vector (IV) is replaced with an explicit IV to protect against CBC attacks [CBCATT].
> - Handling of padding errors is changed to use the bad_record_mac alert rather than the decryption_failed alert to protect against CBC attacks. . . .

EX-1038 pp. 4-5; See also *id.*, p. 21.

> [CBCATT] describes a chosen plaintext attack on TLS that depends on knowing the IV for a record. Previous versions of TLS [TLS1.0] used the CBC residue of the previous record as the IV and therefore enabled this attack. This version uses an explicit IV in order to protect against this attack.

EX-1038 p. 79.

129. TLS 1.2 was still the most up to date version of the protocol by March 2013. EX-1026.

130. As I discussed above, the well-known SSL and TLS protocols insert an encrypted channel running on top of TCP, and underneath HTTP. EX-1096 p.

1. As was well understood prior to March 2013, the SSL/TLS protocols include a higher layer subprotocol sitting on top of the SSL/TLS Record Protocol. There are four different subprotocols depending on the stage of communication between a HTTP/TLS client and HTTP/TLS server (EX-1038 p. 14, EX-1096 p. 3):

- *Handshake* subprotocol: allows the client and server to authenticate each other and to negotiate the cipher suite and other parameters of the connection (including SSL or TLS version). (EX-1038 pp. 31-35)
- *ChangeCipherSpec* subprotocol: makes the previously negotiated parameters effective, so communication becomes encrypted. (EX-1038 pp. 25-26)
- *Alert* subprotocol: used for communicating exceptions and to indicate potential problems that may compromise security. (EX-1038 pp. 26-31.)
- *Application Data* subprotocol: It takes arbitrary application-layer data (*e.g.*, HTTP data) generally, and feeds it through the secure channel. (EX-1038 p. 53)

131. The SSL/TLS Record Protocol first (1) fragments the higher-layer subprotocol data into blocks of 2^{14} bytes or less; then (2) optionally compresses each block of the subprotocol data, (3) adds a Message Authentication Code to

each block, (4) if applicable (for Finished message *Handshake* subprotocol data, and *ChangeCipherSpec*, *Alert*, and *Application Data* subprotocol data), encrypts each block according to the cipher spec, and (5) prepends an SSL/TLS Record header. EX-1038, 14-15, 17-24, 51.

132. As illustrated in the below figure, the SSL/TLS Record header added to every SSL/TLS "Record" (*i.e.* the block formed using the above-described SSL/TLS Record protocol) includes the following three fields: (1) Byte 0: SSL/TLS Record type, (2) Bytes 1-2: SSL/TLS version (major/minor)[4], and (3) Bytes 3-4: Length of data in the Record (excluding the header itself):

[4] In SSL v2 Record Protocol, the SSL Record header has no SSL version (CS-1018) and a POSA familiar with the relevant art would thus know that the version could only be obtained from looking at the unencrypted body (*i.e.* unencrypted SSL Record) of the SSL v2 handshake message packets.

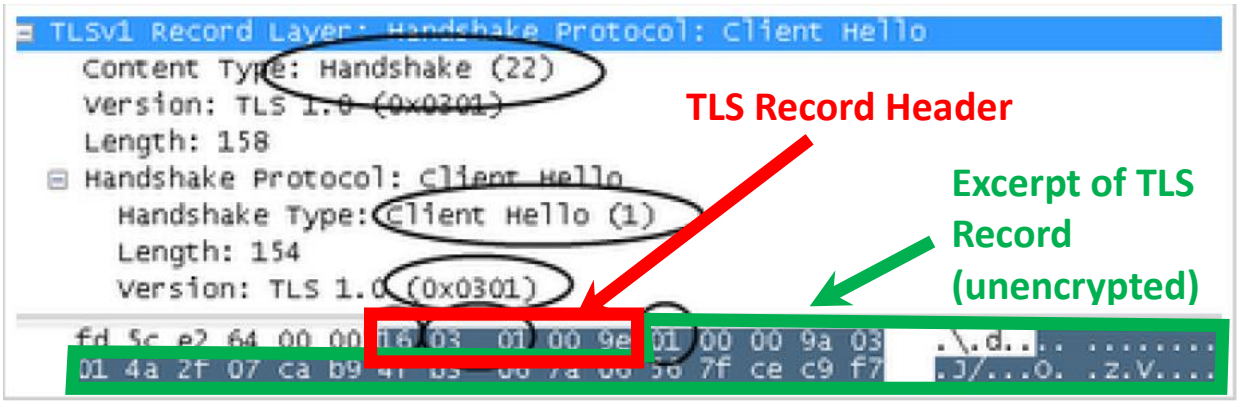
```
        record type (1 byte)
       /
      /    version (1 byte major, 1 byte minor)
     /    /
    /    /         length (2 bytes)
   /    /         /
+----+----+----+----+----+
|    |    |    |    |    |
|    |    |    |    |    | TLS Record header
+----+----+----+----+----+
```

Record Type Values	dec	hex
CHANGE_CIPHER_SPEC	20	0x14
ALERT	21	0x15
HANDSHAKE	22	0x16
APPLICATION_DATA	23	0x17

Version Values	dec	hex
SSL 3.0	3,0	0x0300
TLS 1.0	3,1	0x0301
TLS 1.1	3,2	0x0302
TLS 1.2	3,3	0x0303

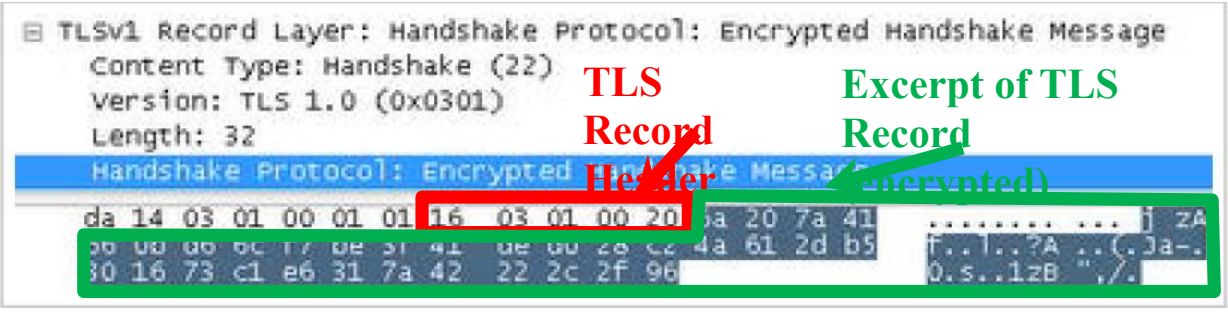
EX-1038, 16-17 (describing header structure illustrated above).

133. A POSA understood that if a user at the HTTP/TLS server (or at an intervening IDS or IPS) examined a TLS Record Protocol packet sent by a HTTP/TLS client requesting to establish a HTTPS session, the packet would include a TLS Record header and an unencrypted TLS Record for a "Client Hello" *handshake* subprotocol message as illustrated in the following example:



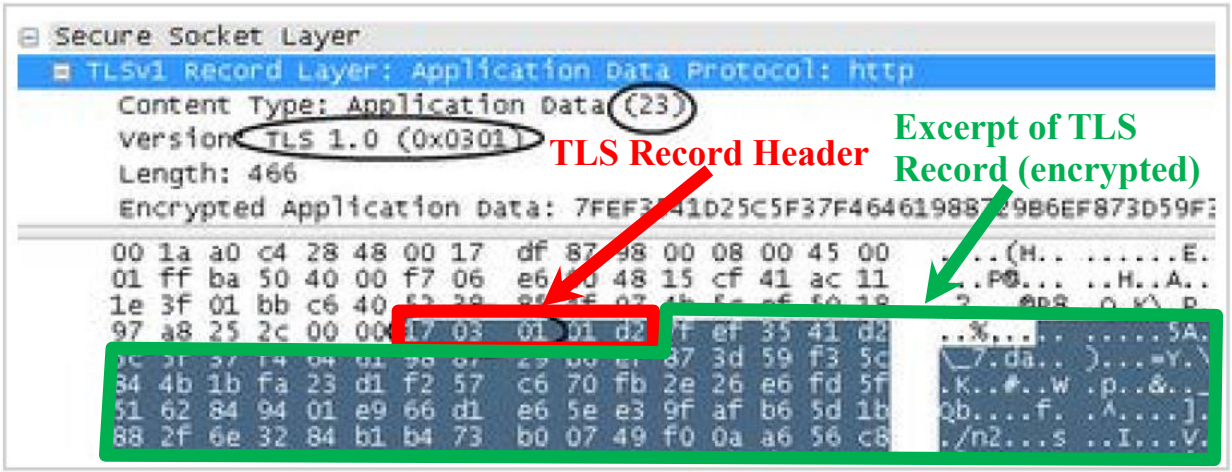
EX-1097 p. 2 (with annotations), EX-1096 p. 2-3;. As shown and annotated in the above example, the TLS Record header specifies that the TLS Record type is Handshake (0x16 or 22), that the TLS version is TLS 1.0 (03x01 or 3,1), and that the length of unencrypted data in the TLS Record is 154 bytes (00x9e).

134. Similarly, a POSA understood that if a user at the HTTP/TLS server (or at an intervening IDS or IPS) examined another TLS Record Protocol packet subsequently sent by the HTTP/TLS client (and as the last message in the *handshake* subprotocol), this packet would include a TLS Record header and an encrypted TLS Record for a "Finished" handshake subprotocol message as illustrated in the following example:



EX-1097 pp. 11-12 (with annotations), EX-1096 pp. 2-3;; EX-1062 p. 62. As shown in the above example, the TLS Record header specifies that the TLS Record type is Handshake (0x16 or 22), that the TLS version is TLS 1.0 (03x01 or 3,1), and that the length of encrypted data in the TLS Record is 32 bytes (00x20).

135. Similarly, a POSA understood that if a user at the HTTP/TLS client (or at an intervening IDS or IPS) examined another TLS Record Protocol packet subsequently sent by the HTTP/TLS server (and as a message in the *application data* subprotocol), this packet would include a TLS Record header and an encrypted TLS Record for an application data subprotocol message as illustrated in the following example:



EX-1097 pp. 12-13 (with annotations), EX-1096 p. 3;. As shown in the above example, the TLS Record header specifies that the TLS Record type is Application

Data (0x17 or 23), that the TLS version is TLS 1.0 (03x01 or 3,1), and that the length of encrypted data in the TLS Record is 466 bytes (01xd2).

136. As identified above, the ’343 Patent describes “TLS Record Protocol packets” as “application packets contained in [] IP packets” and describes the “header fields of TLS Record Protocol packets” as being unencrypted and including “a value indicating the TLS version.” EX-1001 8:14-19.

137. This provides the context for the state of the art prior to March 13th, 2013.

VIII. Claim Construction

138. I have been informed and understand that in order to properly evaluate the ‘343 Patent, the terms of the claims must first be interpreted. I have also been informed and understand that claim terms are given their plain and ordinary meaning as would be understood by one of ordinary skill in the relevant art unless the inventor, as a lexicographer, has set forth a special meaning for a term.

139. I have also been informed that only terms necessary to resolve the controversy need to be construed. In order to construe the claims, I have reviewed the entirety of the ‘343 Patent (EX-1001), its prosecution history (EX-1002), as well as relevant documents from the prosecution and *Inter Partes* Reviews of the antecedent ‘552 and ‘193 patents, and the *Ex Parte* re-examination of the ‘193

patent. The term 'operator' has been a source of confusion in this past history, and thus I discuss it here.

A. "Operator"

140. In mathematics and computer science generally, an "operator" is a function that takes some inputs and produces an output. For example, consider the following definitions from computing dictionaries in my possession:

- Webster's New World Dictionary of Computer Terms, 10th Edition, 2003: "In programming, a code name or symbol that is used to describe a command-name or function, such as multiplying or dividing." (EX-1036.)[5]
- Microsoft Computer Dictionary, 5th Edition, 2002: "n. 1. In mathematics and in programming and computer applications, a symbol or other character indicating an operation that acts on one or more elements." (EX-1037.)
- Minoli-Cordovana's Authoritative Computer and Network Security Dictionary, 2006. Does not have a definition. (EX-1040.)
- McGraw Hill Dictionary of Computing & Communications, 2003. "Anything that designates an operation to be performed, especially the operation code of a computer instruction". (EX-1041.)

[5] Exhibits EX-1036, EX-1037, EX-1040, and EX-1041 contain pages I scanned from my personal copies of these dictionaries.

141. Thus, the term operator can be a very general one, similar to "function" – it designates an entity having a name or a symbol denoting it, and some code or other implementation that takes some operands as input, and produces an output. For example, the '+' operator in its simplest form takes two numbers and produces as output their sum: 2 + 2 = 4. However, many computer languages allow "operator overloading" in which the programmer can create custom definitions of operators over their own data types, and such operators can do anything at all that can be expressed in the computer language.

142. The use of "operator" in the '343 patent is consistent with this general definition. The patent discloses:

> In some variations, multiple packets may be received. A determination may be made that a portion of the packets have packet header field values corresponding to a packet filtering rule. Responsive to such a determination, an operator specified by the packet filtering rule may be applied to the portion of packets having the packet header field values corresponding to the packet filtering rule. A further determination may be made that one or more of the portion of the packets have one or more application header field values corresponding to one or more application header field criteria specified by the operator. Responsive to such a determination, at least one packet transformation function specified by the operator may be applied to the one or more of the portion of the packets.

EX-1001 1:63-2:9. Also:

> For example, such an operator may be able to perform one or more of the following functions: (1) determine if an IP packet traversing a boundary contains an HTTP packet (e.g., an application-level HTTP packet) that specifies one or more specific HTTP methods (e.g., GET, PUT, POST, etc.), and (2) allow the packet (e.g., if a GET method is specified), or block the packet (e.g., if a PUT or POST method is specified). One or more administrators of the secured network may associate such an operator with one or more rules in a network security policy in order to enforce, via the PSG, a Web usage policy that may, for example, allow users to surf (e.g., GET) to one or more web sites attached to the Internet, but may prevent such user(s) from one or more of writing (e.g., PUT) data files or posting (e.g., POST) forms to one or more web sites. For example, administrator(s) may utilize such functionality to prevent one or more exfiltrations (e.g., file transfers containing sensitive information, posting of login credentials (usernames and passwords), etc.) to network nodes (e.g., web sites) that they may not trust.

EX-1001 2:40-58.

143. Thus, the operator in the ‘343 patent is a function which takes as inputs (operands) a series of packets. Packets are themselves a series of eight-bit bytes being transmitted on the network, so as a mathematical object, the operator takes a variable-length sequence of variable-length sequences of bytes as input, and ultimately produces as output a decision as to what to do with the packets

(usually either drop them or forward them). The patent discloses that such operators can look at application header field values within the packets to make the determination. However, other than this, no particular algorithm or structure is proposed for the operators. Rather, it gives a couple of specific examples, expressed in pseudo-code, and teaches that a POSA is expected to use general purpose computing languages to implement the indicated operator functionality:

> One function of operators like HTTP-EXFIL and REQUIRE-TLS-1.1-1.2 may be to inspect fields in the headers of application packets contained in IP packets, determine field values, and depending on the field values, decide to allow, block, or otherwise apply a packet transformation function (e.g., encapsulate the packet into a tunnel, alter one or more header field values, etc.) to the packets. The logic for this function may be executed by one or more of operators **220**, **222**, or **224**. The logic may be developed in a high-level programming language such as C. An example of such programmatic logic, written in pseudocode, for the HTTP-EXFIL operator, is as follows…

EX-1001 7:41-52.

144. I note also that the '343 patent's operators themselves call other functions (since as a minimum they are listed as specifying a "packet transformation function"). In general, it's usually accepted as an implementation detail exactly how a POSA divides a task in their source code into sub-functions.

Since in this case the operator is required to locate the application header field values within the packets, thus requiring parsing both the application layer and the transport and IP layers of the packets, the actual implementation would be quite complex and likely call on a large number of functions (or else access the stored result of many such functions called earlier).

145. In Section IX C below, I detail how the '343 patent's "operator" can be mapped to the Snort rule keyword language, while in the remainder of this section I discuss the history of argumentation over the "operator" term in prior litigation of the '343 patent family.

146. This issue was first addressed in IPR 2018-01436 challenging the '552 patent. There, Patent Owner argued that the board should dismiss the IPR over the petition's and declaration's lack of explicit claim construction of the "operator". EX-1067 p. 14. Patent Owner also argued that Sourcefire did not disclose an equivalent of the '552 patent's "operator". EX-1087 pp. 15-27.

147. However, this putative defect did not prevent the Board from instituting an IPR and ultimately declaring all claims of the '552 patent invalid. First the Board discussed the term "operator" in the claims and summarized:

> Accordingly, the term "operator" as used in the claims is the additional operator described in the Specification that specifies one or

> more application-header-field-value criteria and a packet transformation function.

EX-1067, p. 17. The Board then went on to decide:

> Thus, we agree with Petitioner's argument that Sourcefire discloses an "operator" that specifies (1) the keyword and argument that indicates "application-header- field-value criteria," e.g., TLS version 1.0, and (2) a "packet transformation function," e.g., blocking packets that match the criteria.

(EX-1067, p. 50) and also:

> We are not persuaded by Patent Owner's argument. Instead, for the reasons explained by Petitioner in the Reply, we agree with Petitioner that, as set forth in the Petition, Sourcefire discloses applying an operator in two-stage packet filtering.

EX-1067, p. 52.

148. Patent Owner appealed the Board's decision, but the appeals court confirmed the Board's findings:

> The language of the claims and of Sourcefire provide substantial evidence for the Board's finding that Sourcefire teaches the operator in the '552 patent's claims.

EX-1069 p. 20. The appeals court went onto confirm the Board's finding of the obviousness of all claims. EX-1069 p. 21.

149. In the case of the '193 patent, however, the Board did not agree to institute an IPR, citing the lack of a claim construction, or sufficiently clear

explanation, of the term "operator". EX-1070. The term appears in Claim 1 of that patent, for a representative example. That claim recites:

> …applying by the computing system and to each packet in the first portion of packets, a first operator, specified by the one or more packet-filtering rules, configured to drop packets associated with the particular type of data transfer…

EX-1063, 11:40-44. The Board wrote in its non-institution decision for the '193 related IPR that:

> …it was incumbent upon Petitioner to provide a proposed construction for "operator," or at least to explain what the "plain and ordinary meaning" of the term is in the relevant field. Petitioner's failure to do so weighs in favor of denying the Petition.

EX-1070, p 6. The Board also stated:

> Neither Petitioner nor Dr. Staniford explains sufficiently what aspects of Sourcefire teach a "drop operator" being "applied to the first portion of packets" in any of the three referenced examples.

EX-1070, p 10. The Board thus summarized:

> But therein lies the problem with the Petition as to the first operator limitation—it does not provide adequate explanation. See 37 C.F.R. § 42.104(b)(3)–(5). Consequently, we determine Petitioner has failed to establish a reasonable likelihood of prevailing as to claim 1.

EX-1070, p. 12. The Board did not cite any other reason for denying institution of the IPR than the lack of sufficient claim construction and/or explanation of the "operator".

150. Subsequently, Cisco requested an *Ex Parte* Re-examination of the '193 patent. In those proceedings, Patent Owner argued that the '193 operator was different than the '552 operator (notwithstanding the identical specification of the two patents):

> For the forgoing reasons, and consistent with the Board's Institution Decision in IPR 2018-01559, the "operator" term of the '193 Patent is distinct from the "operator" term claimed in the '552 patent because there is one operator that is "configured to drop packets associated with the particular type of data transfer" and there is a second operator that forwards packets that are not associated with the particular type of data transfer.

EX-1085, p. 127.

> The '193 Patent does not only disclose a simple packet filtering rule that can block all traffic. Rather the '193 Patent introduced the concept of a two-stage packet filtering rule that determines if certain criteria is met, and if so, a separate operator is applied that is configured to drop packets associate with a particular type of data transfer. In other words, the relevant claims of the '193 Patent state that the operator is configured to drop packets associated with a

> particular type of data transfer and there is no such operator in Sourcefire.

EX-1085, p. 128. I will illustrate below in sections IX B and IX D that Sourcefire does indeed teach or disclose such operators (though it doesn't use that word) and that rule writers contributing to the Emerging Threats ruleset achieved exactly equivalent results. For the remainder of this situation, I explain more carefully exactly what the claimed operator is (as the Board suggested I do in my declaration for the '193 IPR).

151. The first important point to note is that the *implementation* of the operator is in the source-code of the system.

> One function of operators like HTTP-EXFIL and REQUIRE-TLS-1.1-1.2 may be to inspect fields in the headers of application packets contained in IP packets, determine field values, and depending on the field values, decide to allow, block, or otherwise apply a packet transformation function (e.g., encapsulate the packet into a tunnel, alter one or more header field values, etc.) to the packets. The logic for this function may be executed by one or more of operators **220**, **222**, or **224**. ***The logic may be developed in a high-level programming language such as C***. An example of such programmatic logic, written in pseudocode, for the HTTP-EXFIL operator, is as follows…

EX-1001, 7:41-52. Emphasis via italics added.

152. The second thing to note is that this functionality in the source code is triggered to run by the presence of text in the rule that denotes the operator. In the examples cited in the prior paragraph, the text is the byte sequence "HTTP-EXFIL" or "REQUIRE-TLS_1.1-1.2" when present in the appropriate place in the rule (e.g. as indicated in the '343 patent's Figure 3 last column).

153. When such text is found in the rule, and the initial five-tuple conditions have been met, then the relevant functionality of the source code is invoked, and that functionality will determine what the relevant application header-field values are, and then accordingly make a decision to drop or allow transmission of the packet(s).

154. Accordingly, the "operator" of the '343 patent is a defined subset or portion of the text in a rule that will invoke the functionality of the source code of the system. The operator may inspect information in packets to determine application header field values in packets seen by the system, and either allow them or drop them (or perform another packet transformation function) according to those values. I will explain exactly how this translates to Snort rules in IX.B below.

IX. Challenge 1: Sourcefire and the Emerging Threats Rules Render Claims 1-20 Obvious

A. Introduction

155. In my opinion, claims 1-20 are obvious in view of Sourcefire 3D System User Guide Version 4.10 ("Sourcefire") in light of the Emerging Threats rules that could be used by Sourcefire. These claims are directed to prior art methods and devices and combinations of conventional components that perform conventional functions regarding rules-based packet filtering, including detecting information exfiltration operations using both packet header fields and application header fields.

156. For example, Claim 1 is directed to a method in which a packet filtering rule has a first criterion that recognizes packet header field values, and if that is matched, then the rule applies a second criterion to application header field values. If that second criterion is also matched, then a packet-transformation function is called to either allow or drop packets with the goal of preventing an exfiltration operation. EX-1001 11:28-61.

157. We can illustrate the claim's steps through Rule 6 of Figure 3:

218

Five-tuple

Rule #	IP Protocol	Source IP Address	Source Port	Destination IP Address	Destination Port	Operator
1 (302)	TCP	140.210.*	*	140.212.*	22	ALLOW
2 (304)	TCP	140.210.*	*	140.212.*	25	ALLOW
3 (306)	TCP	140.210.*	*	140.212.*	110	ALLOW
4 (308)	TCP	140.210.*	*	140.212.*	143	ALLOW
5 (310)	TCP	140.210.*	*	140.212.*	443	REQUIRE-TLS-1.1-1.2
6 (312)	TCP	140.210.*	*	214.*	80	HTTP-EXFIL
7 (314)	*	*	*	*	*	BLOCK

FIG. 3

EX-1001 Fig. 3. The central five columns of the figure (labeled 'Five-tuple') describe the packet header field values that must match:

- the value for the 'protocol' field in the packet IP header must be TCP,
- the IP source address must start with 140.210,
- the destination IP address must start with 214, and
- the destination port must be 80.

158. If this first criterion is met in a given packet (or plurality of packets), then the operator HTTP-EXFIL is applied, which behaves as follows:

> If the application packet contained in the IP packet is an HTTP packet, the value of the HTTP method field in the HTTP packet

> header may be compared to the values that encode the GET, POST, PUT, DELETE, and CONNECT methods. If a match is found, then the HTTP-EXFIL operator may return either ALLOW or BLOCK, depending on the method value.

EX-1001 7:65-8:4. Thus, the test on the HTTP method is the "second criterion" of the claim, which is disclosed to be a test of an application header field value. In my opinion, the HTTP method is an "application header field value". This is confirmed by the pseudocode for the HTTP-EXFIL operator that illustrates GET, POST, PUT, etc. as application header field values. EX-1001 7:52-56.

159. Sourcefire describes the same functionality, and we can see specific examples of very similar rules to that described in the '343 patent's Claim 1 in the Emerging Threats ruleset. For example, consider the following rule (which is SID 2006409 in that ruleset). I will explain the rule syntax in more detail in Section IX C and IX D, but will explain sufficiently to make this example clear here:

```
#alert tcp $HOME_NET any -> $EXTERNAL_NET 81:65535 (msg:"ET
POLICY HTTP POST on unusual Port Possibly Hostile";
flowbits:isnotset,BS.HTTP.ok; flow:established,to_server; content:"POST ";
nocase; depth:5; classtype:policy-violation;
reference:url,doc.emergingthreats.net/2006409;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_HTTP_on_Off_Ports; sid:2006409;
rev:5;)
```

EX-1020, SID 2006409.

160. The first line of the rule above contains the "first criterion" in the '343 patent Claim 1 terminology, or the rule header in the Snort-rule terminology. It reads:

"tcp $HOME_NET any -> $EXTERNAL_NET 81:65535".

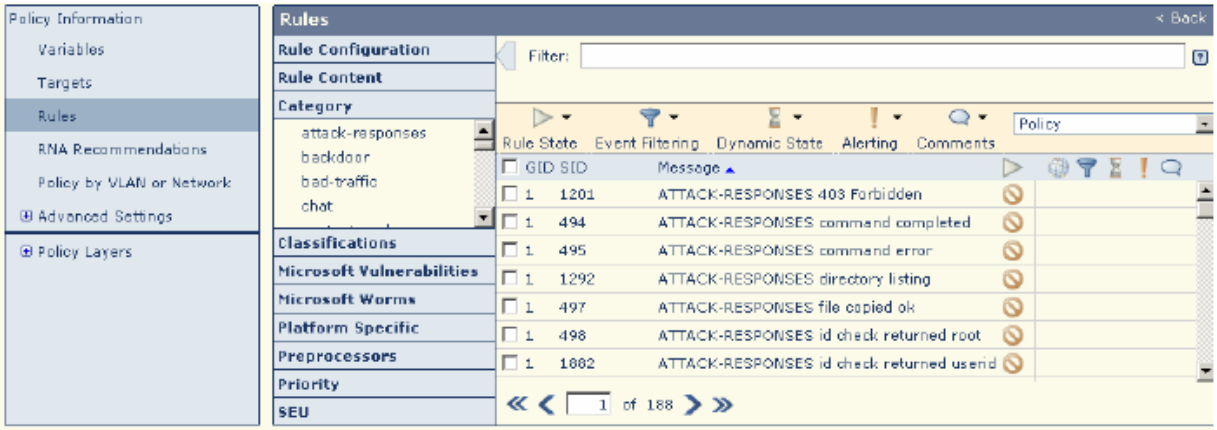
This says that IP packet header protocol value must be TCP (as in the '343 Fig 5 example we are comparing to), that the source addresses must fall in "$HOME_NET" (an address range specified by the user to be their own network), that the destination address must be "$EXTERNAL_NET" (configured to be the network external to the user's home network), and that the destination port must lie between 81 and 65535 – that is, a range of non-standard ports for HTTP traffic to be destined for.

161. If those aspects of the rule match a given packet, then rule 2006409 has additional logic which matches the "second criterion" of Claim 1, specifically that part of the rule that reads:

"content:"POST "; nocase; depth:5;".

This looks for the text being "POST " (including a space after the 'T'), entirely within 5 bytes of the beginning of the HTTP data. I note that the HTTP method is at the very beginning of the request - see Section VII B above for a short HTTP tutorial.

162. Should that criterion match, then the packet transformation function would be applied – also known as the "rule action" in Sourcefire. All Emerging Threats rules are distributed initially with the rule action being "alert," but a POSA would find it obvious that, for any rules they wished to have automatically block traffic on their particular network, that could be achieved by changing the "alert" to "drop" which would cause the packet to be dropped (and subsequent packets could then not be delivered either). Indeed, Sourcefire teaches setting rule states, which allows a user of a Sourcefire Intrusion Protection System (IPS) to change rules from 'alert' (i.e., generate events) to 'drop' (i.e., drop packets that trigger the rules). EX-1004 pp. 435-438. The Rules page, shown excerpted below, lists the rules, and enables a user to select one or more rules and then select the "Rule State" button on the user interface to select whether the rule or rules are set to generate events, drop and generate events, or are disabled.



EX-1004 p. 437. The following excerpt specifically explains how to change the rule state:

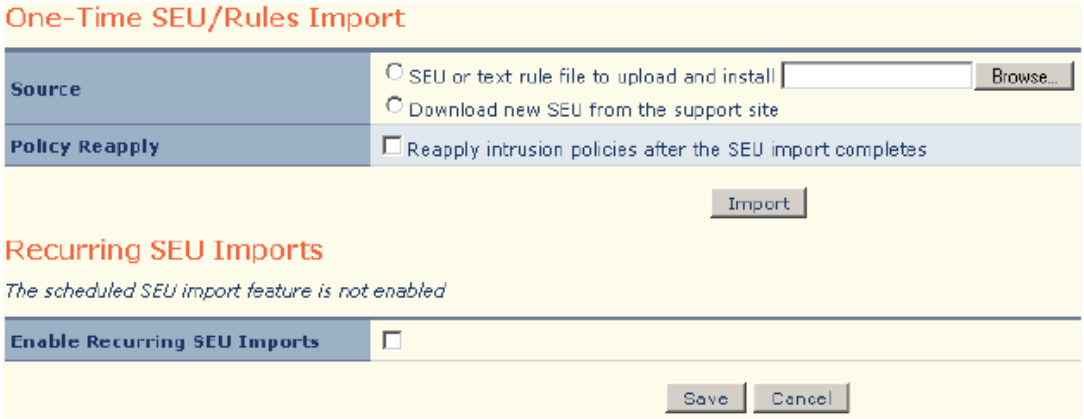
> **5.** Select the rule or rules where you want to set the rule state. You have the following options:
>
> - To select a specific rule, select the check box next to the rule.
> - To select all the rules in the current list, select the check box at the top of the column.
>
> **6.** You have the following options:
>
> - To generate events when traffic matches the selected rules, select **Rule State > Generate Events**.
> - To generate events and drop the traffic in inline deployments when traffic matches the selected rules, select **Rule State > Drop and Generate Events**.
>
> Note that in passive deployments, including when an inline interface is in tap mode, setting the rule state to **Drop and Generate Events** causes rules to generate events without dropping packets.
> - To not inspect traffic matching the selected rules, select **Rule State > Disable**.
>
> **IMPORTANT!** Sourcefire **strongly** recommends that you **do not** enable all the intrusion rules in an intrusion policy. The performance of your sensor is likely to degrade if all the rules are enabled. Instead, tune your rule set to match your network environment as closely as possible.

EX-1004 p. 438. As illustrated in the above excerpt, a user of a Sourcefire system would typically tune their rule set to match his or her specific network environment.

163. Sourcefire also explains that users may import rules from local rule files, which are ASCII text files from a local machine. EX-1004 pp. 1857-1859. Specifically, Sourcefire discloses the following interface screen for importing a text rule file:

2. Click **Import Rules**.

   The Import SEU page appears.



TIP! You can also click **Import SEU** on the SEU page, which you access by selecting **Policy & Response > IPS > SEU**.

3. Select **SEU or text rule file to upload and install** and click **Browse** to navigate to the rule file. Note that all rules uploaded in this manner are saved in the local rule category.

4. Click **Import**.

   The rule file is imported. Make sure you enable the appropriate rules in your intrusion policies. The rules are not activated until the next time you apply the affected policies.

IMPORTANT! 3D Sensors do **not** use the new rule set for inspection until after you apply their intrusion policies. See Applying an Intrusion Policy on page 347 for procedures.

EX-1004 p. 1859. When rules are imported, the Sourcefire IPS sets the rules to a disabled rule state, and requires a user to manually set the state of the local rules before they can be used:

- IPS always sets local rules that you import to the disabled rule state; you must manually set the state of local rules before you can use them in your intrusion policy. See Setting Rule States on page 435 for more information.

EX-1004 p. 1858. As shown above, the portion of Sourcefire explaining how to import rules files expressly explains that once imported, a user of the system would need to select whether rules should provide alerts, or block and provide alerts, and references the Setting Rule States portion of Sourcefire that I addressed in the paragraph above. This shows that it was customary for users of Sourcefire to import custom rules, and then to select whether the rules would be set to just alert, or to block traffic.

164. As I will describe in more detail below, Emerging Threats was a well-known source of such custom rules. I used Snort rules, including custom Snort rules, extensively in the late 1990s and early 2000s and was familiar with Bleeding Snort, which later became Emerging Threats. A POSA by 2010 would have been familiar with Emerging Threats as a source of custom rules, and would have downloaded text (ASCII) files of rules from Emerging Threats to load into Snort devices, such as Sourcefire. As addressed in the prior two paragraphs, Sourcefire would disable the rules by default upon their import, and a user would then select whether to set the rules to alert or drop. A user would often initially set imported rules to have an "alert" action to evaluate their performance in the user's specific

network environment. This would avoid inadvertently blocking legitimate traffic if the rule were to generate a significant volume of false positives. If the alerts showed that the rule operated as desired in the applicable network environment, the user would then follow the instructions in Sourcefire to have the rule drop traffic, as described above.

165. The Snort rule message documents the purpose of the rule for humans as "ET POLICY HTTP POST on unusual Port Possibly Hostile", making the suggestion to the rule user that if they considered POSTs of data to non-standard HTTP ports as hostile transmissions of data, they were invited to utilize this rule to detect or prevent such transfers as a matter of "POLICY". This would inform the user that they could change the rule to block the exfiltration attempts, as explained immediately above. Accordingly, the rule was capable of blocking exfiltrations using an essentially identical technique to that disclosed in the '343 patent.

166. I will take up the claims in more detail below in Section IX E after introducing the Sourcefire reference and the Emerging Threats ruleset in the intervening sections.

B. The Sourcefire Reference (EX-1004)

167. The Sourcefire reference is a user manual titled "Sourcefire 3D System User Guide" Version 4.10 ("Sourcefire"). The Sourcefire system was a commercially successful IPS with a particularly detailed manual, and thus it serves

to document that features claimed in the '343 patent were in fact well known some years earlier. I am aware of these facts as, during the time I was running Silicon Defense, I directly competed with Sourcefire (the company) and thus paid close attention to the early development of that firm.

168. I have reviewed the Declarations of John Leone (EX-1005) and Jacob H. Baugher III (EX-1046) and understand that Sourcefire was distributed and otherwise made publicly accessible at least as early as April 2011 as it was enclosed on documentation disks (CD-ROM or DVD) included with each Sourcefire 3D System appliance subsequently sold (e.g., 3D Sensor, Defense Center) by Sourcefire, Inc. EX-1004. Based on these declarations, I also understand that Sourcefire was distributed and otherwise made publicly accessible at least as early as April 2011 as it was publicly accessible for download by persons who had received a login and password from Sourcefire, Inc. to its support website (and specifically for the website at the uniform resource locator ("URL") of https://support.sourcefire.com). EX-1004; EX-1005; EX-1033 (Sourcefire Support Website); EX-1045. I understand that Sourcefire (EX-1004) was distributed or available to over 500 customers between April 2011 through March 2013, which included entities having large network security teams, such as Microsoft and various U.S. government agencies. EX-1005 ¶¶18-19, EX-1046 ¶9.

This is consistent with my recollection of Sourcefire being well-known during that timeframe.

169. I have been informed and understand that Sourcefire is a printed publication and is available as prior art under pre-AIA 35 U.S.C. § 102(b). I have further been informed and understand that the Board concluded that Sourcefire is a printed publication that constitutes prior art to the '552 patent (from which the '343 patent claims priority) under § 102(b) in their final written decision in IPR2018-01436 (of the '552 patent), and the Federal Circuit affirmed the Board's finding on appeal. EX-1067 p. 31; EX-1069 pp. 13-17.

170. Sourcefire is a user manual for the Sourcefire 3D System. The 3D System could identify changing assets and vulnerabilities on the network, determine the types of attacks against the network and their impact, and defend the network in real time. EX-1004 p. 32.

171. Sourcefire describes packet-filtering devices (3D Sensors) of the 3D System that a user (e.g., a network security administrator) deployed and configured in a network to either passively or inline monitor network traffic. Each 3D Sensor deployed in a network was capable of running any combination of three major software components: (1) Intrusion Protection System (IPS), (2) Real-time Network Awareness (RNA), and (3) Real-time User Awareness (RUA). The IPS

provided "best-in-class intrusion detection and prevention by harnessing the power of the industry-standard Snort® rules-based detection engine. Backed by the acclaimed Sourcefire Vulnerability Research Team™ (VRT), Sourcefire delivers its customers unrivaled protection against known and unknown threats." EX-1004 pp. 33-34.

172. Sourcefire discloses that the 3D Sensor running the IPS monitored the network for attacks that might affect the availability, integrity, or confidentiality of hosts on the network. Each deployed 3D Sensor with IPS examined the packets that traversed a network, looked for malicious activity, and reported and/or took actions based on detected malicious activity. The IPS performed these functions on packets using a series of decoders, preprocessors, and a rules engine (as illustrated below), each of which implemented rules to look for the broad range of malicious behaviors that attackers had developed. EX-1004 pp. 35, 254-256.

173. The rules engine of the IPS implemented intrusion rules to determine whether the headers and/or payloads of received packets triggered one or more of such rules. EX-1004 pp. 256-259, 513, 762-767, 2084, 2089. The 3D Sensor with IPS may be deployed in an inline detection mode to drop or replace packets that the various rules stored on the 3D Sensor detected as being harmful. EX-1004 pp. 262-265, 276-277, 761.

174. The rules supplied with a 3D Sensor with IPS were developed by the Sourcefire VRT. These rules (implemented by the code of the decoders and preprocessors, or in the rules engine) were selectable by a user to detect the most likely attacks to occur on such user's network. IPS allowed a user to modify various parameters of these rules to customize the rules as appropriate or desired for a network and to write their own custom intrusion rules tuned to the user's specific network environment and the types of anomalies and malicious activities that such environment is likely to encounter. EX-1004 pp. 256-260, 428-430, 761-770.

175. The intrusion rules had 5-tuple values associated with them. EX-1004 pp. 256-259, 761-764. Intrusion rules[6] contained two logical parts: the rule header and the rule options. The rule header part contained:

- the 5-tuple:

[6] I use "intrusion rules" herein to refer to "standard text" rules as opposed to "shared object" rules which included customizable "rule header" parts but not customizable "rule options" parts. EX-1004 pp. 405-406, 761-771. Although Sourcefire uses the term "intrusion rules" for convenience, the rules were used for a number of purposes, including exfiltration detection, as discussed below.

    - the protocol;
    - the source and destination IP addresses and netmasks;
    - the source and destination ports;
- the rule's action or type (e.g. alert and allow, drop, ignore and allow); and
- direction indicators showing the flow of traffic from source to destination.

The rule options part contained:

- event messages;
- keywords and their parameters and arguments;
- patterns that a packet's payload must match to trigger the rule (including application header field values); and
- specifications of which parts of the packet the rules engine should inspect.

EX-1004 pp. 358, 488-490, 761-766.

176. An example intrusion rule in Sourcefire illustrates these different parts of a rule:

Rule Header

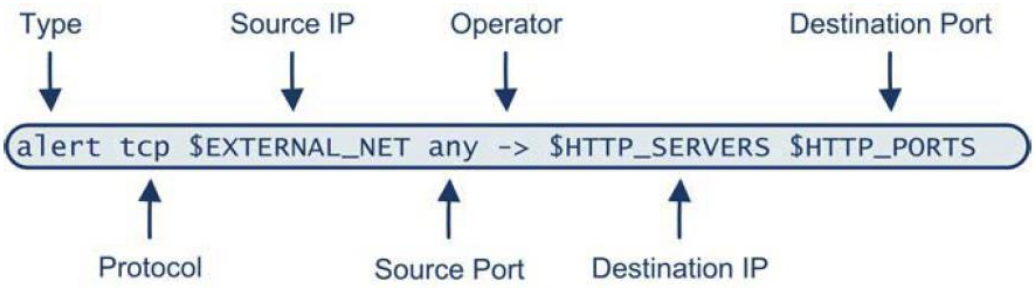
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
```

Rule Keywords and Arguments

```
(msg:"WEB-IIS newdsn.exe access";
flow:to_server,established; uricontent:"/scripts/
tools/newdsn.exe"; nocase; metadata:service http;
reference:bugtraq,1818; reference:cve,1999-0191;
reference:nessus,10360; classtype:web-application-
activity; sid:1024; rev:10; )
```

EX-1004 pp. 488, 762-763. This rule alerted the user based on TCP packets from any external network source IP address and port, destination IP addresses and ports matching known values for HTTP servers, and including a URI (/scripts/tools/newdsn.exe) used in a popular attack which matched the corresponding content keyword. *Id.* The rule also identified its message, unique identifying number (Snort ID or SID 1024) and the various sources (identified by the "reference" text) to external reports about the exploit the rule detected. EX-1004 pp. 776, 870, 873-875, 2093.

177. Sourcefire explains that every intrusion rule had parameters and arguments that made up the rule header (including the 5-tuple). For example, the following figure and table illustrate the parts of a rule header in the example intrusion rule and how they could be used by IPS:



EX-1004 p. 764.

Rule Header Values

Rule Header Component	Example Value	This Value...
Action	`alert`	Generates an intrusion event when triggered.
Protocol	`tcp`	Tests TCP traffic only.
Source IP Address	`$EXTERNAL_NET`	Tests traffic coming from any host that is not on your internal network.
Source Ports	`any`	Tests traffic coming from any port on the originating host.
Operator	`->`	Tests external traffic (destined for the web servers on your network).
Destination IP Address	`$HTTP_SERVERS`	Tests traffic to be delivered to any host specified as a web server on your internal network.
Destination Ports	`$HTTP_PORTS`	Tests traffic delivered to an HTTP port on your internal network.

EX-1004 pp. 762-764.

178. Sourcefire discloses that the rules could be applied on a packet-by-packet basis and that the type of actions that could be triggered by the rules were *pass* (and ignore), *alert* (and pass) or *drop* (block). EX-1004 pp. 256-260, 358-359, 762, 764-769.

179. The IPS in each 3D Sensor was designed such that packets would trigger a rule when data within a packet satisfied one or more specified "criteria"

that corresponded to one or more of a plurality of packet header field values (e.g. source or destination IP address, source or destination port, and IP protocol) EX-1004 pp. 762-769.

180. Sourcefire also discloses that, based on keywords in these intrusion rules, packets were filtered as they arrived at various application layer preprocessors and at the rules engine based on various application layer data in such packets including data associated with particular types of data transfer and data associated with instant messaging applications. EX-1004 pp. 334, 373, 428-429, 435-439, 491, 538-542, 546-560, 597-601, 697-701, 765, 785-786, 827-828, 874, 2032-2043, 2092.

181. Sourcefire describes that a user could configure various application layer preprocessor rules and intrusion rules to only look for packets traveling over standard ports for a particular application layer protocol (*e.g.*, TCP ports 80 for HTTP, 21 for FTP, 25 for SMTP, 443 for HTTPS, etc.) or could configure such rules to be "adaptive" to identify particular application layer protocol packets traveling over non-standard ports. With adaptive profiles enabled in such rules, Sourcefire describes that the applicable application layer preprocessor (*e.g.*, HTTP, FTP, SSL) and/or rules engine checked the packets for particular application layer protocol identifiers to determine if the packets were traffic for the corresponding

application layer protocol (*e.g.*, HTTP, FTP, SSL/TLS Record Protocol). If an identifier is found for a particular application layer protocol, the corresponding preprocessor was invoked to process the now-identified packets using the keyword(s) and arguments of the corresponding preprocessor rules and intrusion rules even if the packets came over a non-standard port for such protocol. EX-1004 pp. 598, 697-701.

182. For example, Sourcefire describes that keywords of intrusion rules could be used by the application-layer preprocessor called the SSL preprocessor and rules engine of a 3D Sensor to filter packets by encryption protocol version (*e.g.* Transport Layer Security (TLS) version or TLS's predecessor Secure Sockets Layer (SSL) version). For example, the ssl_version keyword could be used in an intrusion rule, causing the SSL preprocessor to match against such protocol version information in the application-layer header (*e.g.*, SSL/TLS Record header) of received packets and/or unencrypted application layer payload (*e.g.*, unencrypted SSL/TLS Record) of received handshake message packets for an encrypted session.

183. As another example, Sourcefire discloses that HTTP content keywords of preprocessor rules and intrusion rules, including in the "adaptive" mode, could be used by another application-layer preprocessor called the HTTP

inspect preprocessor and rules engine of a 3D Sensor to search for matched application layer content in packets such as known HTTP methods "GET, PUT, CONNECT, and so on" and *pass* (allow), *alert* (and pass), or *drop* (block) packets based on results of such searches. EX-1004 pp. 785-786, 700. Thus, when a packet triggered the rule, the IPS applied the "packet transformation function" specified by the rule (*e.g*. pass, drop, alert (and pass)) to the triggering packet. EX-1004 pp. 435-439, 295-297, 765. Said another way using the terminology of the '343 patent, Sourcefire discloses applying the "packet transformation function" specified by the operator to packets if their application header field values (e.g., the HTTP methods) correspond to application header field criteria specified by the operator (e.g., GET, PUT, etc.).

184. As another example, Sourcefire discloses that multimedia application content keywords could be used by the rules engine of a 3D Sensor to search for matched application-layer content in packets, for example, to enforce a security policy prohibiting employees' use of such applications, and *pass* (allow), *alert* (and pass), or *drop* (block) packets based on results of such searches. EX-1004 pp. 266, 333-335, 429-430, 874, 1453, 1470-1471, 2034, 2037, 2039, 2040-2042, 2092; Section VII.D.

185. In summary, Sourcefire described a very flexible approach to customizing Sourcefire-provided intrusion rules, and writing new intrusion rules (including using Sourcefire-provided keywords), which enabled users to implement many different types of intrusion rules in their particular intrusion policy in their particular IPS on received packets that, when certain rules triggered, some packets were allowed to pass (with or without an alert), and, when other rules triggered, blocked other packets by dropping them.

C. The '343 "Operator" as expressed in Sourcefire's Snort Rules

186. The '343 patent does not disclose any definite algorithm, architecture, or language for implementing the "operators" that it discloses. Rather, it gives two particular examples, expressed in pseudo-code, and teaches that a POSA is expected to use general purpose computing languages to implement the indicated operator functionality:

> One function of operators like HTTP-EXFIL and REQUIRE-TLS-1.1-1.2 may be to inspect fields in the headers of application packets contained in IP packets, determine field values, and depending on the field values, decide to allow, block, or otherwise apply a packet transformation function (e.g., encapsulate the packet into a tunnel, alter one or more header field values, etc.) to the packets. The logic for this function may be executed by one or more of operators **220**, **222**, or **224**. The logic may be developed in a high-level programming language such as C. An example of such

> programmatic logic, written in pseudocode, for the HTTP-EXFIL operator, is as follows…

EX-1001 7:41-52.

187. In this section of the document, I explicitly spell out how the functionality of Snort rules was used to implement what the '343 patent later referred to as its operator. As a fairly general mechanism for identifying and matching data in packets, the Snort rule keywords are amply expressive enough to cover a large number of application header fields, and certainly all the examples disclosed in the '343 patent. Like the '343 operator, Snort rule keywords invoked particular functionality in the code of the Sourcefire system to perform their specific functionality. Later in this section, and more so in the next section, I will cover specific examples in which '343 operator functionality is clearly present in particular Snort rules that were publicly available long before March 2013.

188. The basic structure of the '343 packet filtering process, with a "first criterion" in a rule distinguishing based on packet headers, and a "second criterion" in the rule distinguishing based on application-layer headers mirrors how rules are processed in the Snort detection engine that underpins Sourcefire. As explained in the original paper that introduced the system, rules are divided into two parts: a "Chain Header" that considered the most important packet header fields such as IP

addresses and TCP/UDP ports, and then a "Chain Option" that considered everything else, and in particular, all consideration of the application layer:

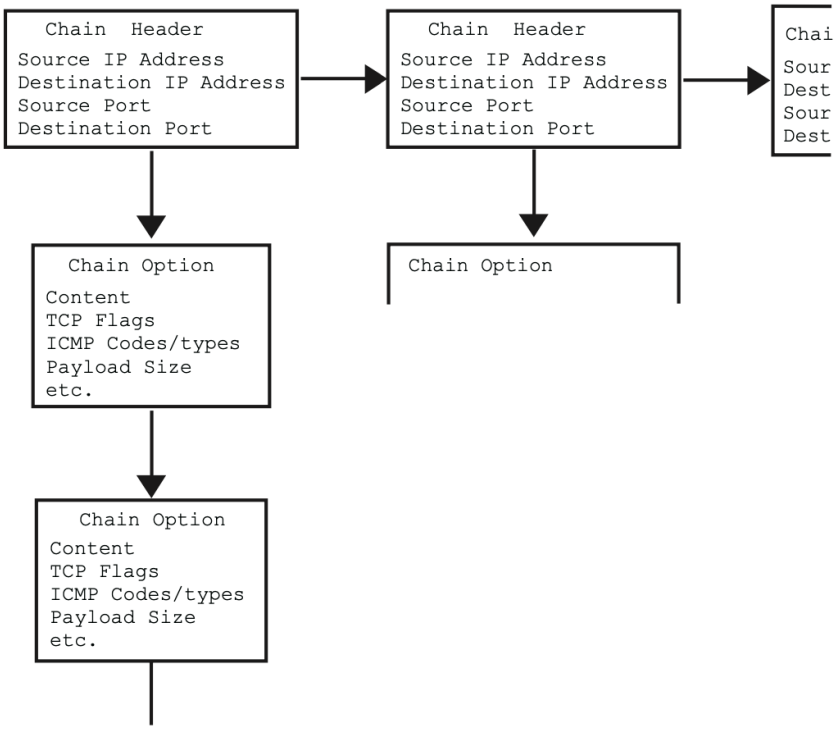


Figure 3: Rule Chain logical structure.

EX-1018 Fig. 3. The paper went on to explain:

> Snort maintains its detection rules in a two dimensional linked list of what are termed Chain Headers and Chain Options. These are lists of rules that have been condensed down to a list of common attributes in the Chain Headers, with the detection modifier options contained in the Chain Options. For example, if forty-five CGI-BIN probe detection rules are specified in a given Snort detection library file, they generally all share common source and destination IP addresses and ports. To speed the detection processing, these commonalities are

> condensed into a single Chain Header and then individual detection signatures are kept in Chain Option structures.

EX-1018 p. 4. A POSA would understand that the point of this data-structure is that the list of Chain Options attached to a given Chain Header is not checked, for a given packet, unless the 5-tuple conditions in the Chain Header have first been matched for that packet:

> To speed the detection processing, these commonalities are condensed into a single Chain Header and then individual detection signatures are kept in Chain Option structures.

EX-1018 p. 4.

189. The '343 patent discloses that the operator specifies a "Packet Transformation Function". The clearest delineation of this is as follows:

> One function of operators like HTTP-EXFIL and REQUIRE-TLS-1.1-1.2 may be to inspect fields in the headers of application packets contained in IP packets, determine field values, and depending on the field values, decide to allow, block, or otherwise apply a packet transformation function (e.g., encapsulate the packet into a tunnel, alter one or more header field values, etc.) to the packets.

EX-1001 7:41-48.

190. This functionality corresponds to the "Rule Action" disclosed in Sourcefire. EX-1004 pp 765. After the rule header and the rule options have been evaluated, if the rule matches the packets, then the rule action is taken on those

packets. Sourcefire discloses three rule actions: *drop* (corresponding to the BLOCK transformation function in the patent), *alert* (which allows the packet through but generates an event recording the rule match for further analysis by system users, and *pass* (which allows packets through with no record of them created, and is used for whitelist rules). EX-1004 pp. 359, 435, 761-765, 863, 2075, 2079, 2087.

191. In my opinion, the Sourcefire disclosed *replace* keyword also falls in the scope of the '343 patent's "packet transformation function", as this keyword allows a rule that matches some content in the packet(s) to replace that with some other content (e.g. a possible exploit to be replaced with innocent data, or revealing confidential information to be replaced with innocuous data). EX-1004 pp 792-793.

192. At times, the patent uses the term "operator" to just refer to the action taken on the packet(s). For example:

> Rule 7 314 may be a "wildcard" rule and may apply a BLOCK operator (e.g., a null operator which "drops" any packets it is applied to) to any packets that do not match the criteria of any of Rules 1 302, 2 304, 3 306, 4 308, 5 310, or 6 312…

EX 1001 6:34-38. This appears to be the situation in which the operator has no specific application layer detection logic, and instead just specifies a packet

transformation function that is applied to *all* packets on which the operator is invoked.

193. However, in the '343 patent's terminology, operators can have application layer logic in addition to the packet transformation function (rule action), and which determines which packets the transformation function is applied to:

> …the network packet filter may implement one or more additional operators having the capability to determine if a packet contains an application-level header that specifies a particular method associated with a data transfer protocol; and, if so, whether to apply an identity operator or null operator to the packet. To distinguish a network packet filter that implements said additional operators from network packet filters that do not, a network packet filter that implements such additional operators will be referred to hereinafter as a packet security gateway (PSG).

EX-1001 2:28-39.

194. Sourcefire has a rich language allowing rule-writers to specify the "operator" part of the rule that checks application-level headers. In particular, the following keywords allow specification of equivalent functionality to the '343 patent's "operator":

- *content* (EX-1004 pp. 776-777) – this matches arbitrary byte strings in the application layer traffic and is often used to match application header values.

- *nocase* (EX-1004 p. 778) – specifies that a content match is case-insensitive.
- *pcre* (EX-1004 pp. 800-809) – matches arbitrary regular expressions in the application data and can be used to match application header field values.
- *not* (EX-1004 p. 779) – negates a content match.
- *rawbytes* (EX-1004 p. 779) – matches in the raw application data prior to any normalizations applied by any Sourcefire application preprocessors.
- *distance* (EX-1004 p. 781) – minimum number of bytes for a content match to follow the prior content match. This can be used to constrain where a content match can take place.
- *within* (EX-1004 pp. 781-782) – maximum number of bytes for a content match to follow the prior content match. Again, this can constrain where in the packet or data-stream a content match can take place.
- *offset* (EX-1004 p. 782) – minimum number of bytes for a content match to appear from the beginning of the application payload. Constrains content match locations.
- *depth* (EX-1004 p. 783) – maximum number of bytes for a content match to appear from the beginning of the application payload. Constrains content match locations

- *http_client_body* (EX-1004 pp. 783-785, 788) – this modifies content matches to specifically search within the body of an HTTP request (if appropriate options are set for the HTTP preprocessor). This might or might not be an "application header" depending on whether there is another application being channeled over the HTTP application layer.
- *http_cookie* (EX-1004 pp. 783-785, 787) – modifies content matches to search in HTTP Cookie headers (after normalization by the HTTP preprocessor). The values of HTTP cookies are definitely "application header field values", with the application being HTTP, so any appearance of this in a rule will match that aspect of the '343 operator.
- *http_raw_cookie* (EX-1004 pp. 783-785, 787) – modifies content matches to search in HTTP Cookie headers (before normalization by the HTTP preprocessor). The values of HTTP cookies are definitely "application header field values", with the application being HTTP.
- *http_header* (EX-1004 pp. 783-785, 786) – modifies content searches to match within the values of HTTP header fields (i.e. after the colon) after normalization. By definition such matches are matching "application header field values."

- *http_raw_header* (EX-1004 pp. 783-785, 786) – modifies content searches to match within the values of HTTP header fields (i.e. after the colon) before normalization. By definition, such matches are against "application header field values."
- *http_method* (EX-1004 pp. 783-785, 786) – matches in the value of the HTTP method of an HTTP request. This allows matches akin to the HTTP-EXFIL operator in the patent. EX-1001 7:51-8:6.
- *http_uri* (EX-1004 pp. 783-785, 785) – modifies content searches to match within the uniform resource identifier in an HTTP request after normalization. The URI of an HTTP request is also an "application header field value".
- *http_raw_uri* (EX-1004 pp. 783-785, 786) – modifies content searches to match within the uniform resource identifier in an HTTP request before normalization. The URI of an HTTP request is an "application header field value".
- *http_stat_code* (EX-1004 pp. 783-785, 788) – modifies content searches to match the 3-digit numerical status code in an HTTP response (which is an application header field value).

- *http_stat_msg* (EX-1004 pp. 783-785, 788) – modifies content searches to match the ASCII text status message in an HTTP response (which is also an application header field value).
- *uricontent* (EX-1004 p. 831) – equivalent to a 'content' followed by a http_uri (and thus matches an application header field value).
- *urilen* (EX-1004 pp. 830-831) – matches against URI's in HTTP request of particular lengths or ranges of lengths (in bytes).
- *isdataat* (EX-1004 pp. 837-838) – constrains a content match to be at a particular offset to the beginning of the application data, or a prior offset. This can be used to constrain matches to be particular application field values in some cases.
- *byte_jump* (EX-1004 pp. 794-797) – Compute a derived position with the application data to test content, based on values obtained from that data, rather than specified in advance.
- *byte_test* (EX-1004 pp. 797-800) – Check whether values calculated from packet data match and expected value or range.

195. An example will help to make the situation clearer. We pick the following outbound HTTP spyware connection rule:

```
#by Matt Jonkman
#
```

```
alert tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS
(msg:"ET MALWARE Rabio Spyware/Adware Initial
Registration"; flow:established,to_server; dsize:<200;
content:"POST "; depth:5; content:"|0d 0a 0d
0a|REGISTER|7c|";
pcre:"/REGISTER\x7c\d+\x7c\d+\x7c\d+\x7c\d+/";
classtype:trojan-activity;
reference:url,www.spywareguide.com/product_show.php?id=
3770; reference:url,www.rabio.com;
reference:url,doc.emergingthreats.net/bin/view/Main/200
7820; reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/MALWARE/MALWARE_Rabio; sid:2007820;
rev:2;)
```

EX-1020, SID 2007820. Here, the rule header "`tcp $HOME_NET any -> $EXTERNAL_NET $HTTP_PORTS`" and "`flow:established,to_server`" establishes that the operator will only be looking at application layer payload going to HTTP ports – that is, in the common case, HTTP requests. This is the packet header value test (first stage) in the '343 patent's two stage process. There are then four components to the application-layer inspection in the rule options:

- "`content:"POST "; depth 5`". This checks that the HTTP method of the request is POST (note the space after "POST" and that this must occur within the first 5 bytes due to the depth statement in the operator). This is akin to the HTTP example in the '343 patent and is clearly checking an

"application layer header value", and thus is disclosing the use of operator functionality. EX-1001 2:40-58.

- "`content:"|0d 0a 0d 0a|REGISTER|7c|";`" . This checks for a "REGISTER|" operation in the web application of the malware (at the beginning of the HTTP request body).
- "`dsize:<200`" tests that the payload size is less than 200 bytes.
- "`pcre:"/REGISTER\x7c\d+\x7c\d+\x7c\d+\x7c\d+/";`". Checks that the REGISTER command is followed by a correct sequence of numbers separated by pipe ('|') characters.

At least the 'content: "POST"' checking of the http_method is unambiguously operator functionality because it is identical to the functionality disclosed in HTTP-EXFIL in the '343 patent. A POSA would understand that the content and PCRE checks for 'REGISTER' are also checking application layer header functionality of the malware application running over HTTP.

196. Note that not all option keywords disclose functionality equivalent to the operator in the patent. In particular, some keywords serve the purpose of documentation or specifying metadata for the rule. Examples of these keywords would be *msg* (EX-1004 p. 772), *reference* (EX-1004 p. 776), *sid* (EX-1004 p. 861-862), *classtype* (EX-1004 pp. 772-775), *priority* (EX-1004 p. 772), *metadata*

(EX-1004 pp. 810-812). Others are performance related without direct impact on the rule semantics (e.g. *fast_pattern* (EX-1004 pp. 789-792)).

197. Similarly, some functionality in Snort rules, such as content matching, can be used, and frequently has been used, for detecting "application layer header values". However, the Snort content matching can be used to detect any content at all in network packets, including, in applications that have a clear distinction between the "header" and the "body", content deep within the body. For example, Snort content matching can be used to detect particular text strings deep in the text of a web page, which might not reasonably be considered an "application header field value."

198. However, the fact that Snort rule options have non-operator functionality does not preclude that Sourcefire (and extant Snort rules) disclose operator functionality; the '343 patent teaches that operators can be implemented in general purpose computing languages such as C, and those languages have many uses and functionalities other than checking application layer header field values in network packets.

199. In its decision not to institute an IPR of the '193 patent, the Board raised the concern that:

> The Specification of the '193 Patent also indicates that the recited operators may be distinct from the recited packet-filtering rules. For

> example, the Specification indicates that some packet-filtering rules are associated with an operator, but others are not. *See* Ex. 1001, 2:16–19 (stating that one or more rules "may" be associated with an operator). In addition, an operator may be associated with multiple rules.

EX-1070 p. 11. That concern does not apply to Sourefire in view of Emerging Threats discussed below. For example, consider the example above of 'content "POST "' in the context of drop rules. This is an operator in the sense that it is a subset of the text of the rule that invokes the source code of the system to check for a particular application header field value, and then results in dropping the packets in that connection, or not, based on that value. I checked the Emerging Threats rules, and of the 2797 rules involving "$EXTERNAL_NET $HTTP_PORTS" (i.e., concerning traffic with remote HTTP servers) the operator 'content "POST "' occurs 148 times, while 'content:"GET "' occurs 284 times. Thus, these operators occur many times, and are thus "associated with multiple rules". Of course, the Emerging Threats rules disclose thousands of different operators, whereas the '343 patent only details two. But this reflects that Sourcefire/Emerging Threats describe a large real-world working system developed by hundreds of people over more than a decade, whereas the '343 patent was the work of a single inventor who shows very little awareness of the prior art.

200. Similarly, it is entirely possible that "some packet-filtering rules are associated with an operator, but others are not", in that there are numerous Emerging Threats rules that have no application layer logic but make the decision what to do entirely based on packet header fields. For example, consider this rule:

```
#alert tcp $EXTERNAL_NET any -> $HOME_NET any
(msg:"ET EXPLOIT Invalid non-fragmented packet with
fragment offset>0"; fragbits: !M; fragoffset: >0; classtype: bad-
unknown;
reference:url,doc.emergingthreats.net/bin/view/Main/2001022;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/EXPLOIT/EXPLOIT_Invalid_TCP_Fragmen
ts; sid:2001022; rev:5;)
```

EX-1020 SID 2001022. This is using the "fragbits" and "fragoffset" keywords to look for "Invalid non-fragmented packet with fragment offset>0" in TCP packets. EX-1004 pp. 813, 839. There are no potential operator keywords looking at the application content; the decision to drop, alert, etc. (depending on the packet transformation function specified by the rule action) will be made entirely on information in the IP header.

201. What is important is that these references show that Snort rules were used to detect application layer header field values in service of detecting or blocking exfiltrations, that they were specifically documented as being designed to perform this in numerous instances, and that actual Snort rules doing so were widely available, a subject to which I turn next.

D. The Emerging Threats Rules

202. The core NIDS engine in Sourcefire, Snort, was begun as an open source project in the late 1990s. EX-1018. From the beginning, the rulesets were open-source and accepted community contributions. Further, besides rules from the Snort.org website (which was run by Sourcefire once the latter incorporated), there were independent rule providers. For example, the Bleeding Snort project launched in 2004 to be an independent clearing house for Snort rules to detect a variety of problems. EX-1050, EX-1051. A contemporary description of the project at the time of the launch reported:

> A new Web site, The Bleeding Edge of Snort, is available for those who use Snort IDS. Among other things, the site offers new Snort rules that aren't necessarily part of the regular Snort rules distribution.
>
> For example, there are intrusion detection rules for the issues that were recently discovered in ASP.NET. There are also rules to detect certain denial of service (DoS) attacks against Cisco routers, DoS attacks against Microsoft SMS, buffer overflows in Adobe Acrobat Reader, almost two dozen rules that detect exploits against Internet Explorer (IE), over a dozen rules to detect exploits against Internet Information Server (IIS), rules for Microsoft SQL Server exploits, rules to help protect against various file downloads such as registry files, executables, DLLs, and a much more. The Snort rules are freely available to the public and new rules can be contributed by anyone.

EX-1050.

203. The project continues today, renamed to Emerging Threats and sponsored by Proofpoint. EX-1053, EX-1054. The project Frequently Asked Questions document says:

> Emerging Threats is a division of Proofpoint, Inc. Our primary projects are the Emerging Threats Ruleset, contributed and maintained by the security community, and the Emerging Threats Pro Ruleset, which is maintained by the Proofpoint/ET research team.

and:

> Are the Emerging Threats Rules REALLY free? Yup. Free, as in BSD licensed, which allows you to do what you like with them. All we ask is that when you have an idea, a new signature, or even just a theory, that you send it in to benefit everyone.

EX-1078.

204. Via the Archive.org Wayback machine, I obtained a copy of the Emerging Threats rules as they were archived on December 2, 2010 (i.e. more than a year before the March 12, 2013 earliest priority date for the '343 patent). EX-1020. Emerging Threats is distributed as a text file of Snort rules. I analyzed many of the rules in Emerging Threats, including those discussed in this declaration, and based on my experience the Emerging Threats rules appear to be true copies of rules that existed in 2010.

205. The file consists of 11,443 Snort alert rules (including rules commented out or commented as marked for deletion). Rules are generally labeled with the author who contributed that particular rule, and my review of the file revealed several hundred different rule authors. In my opinion, the file represents the summation of a major effort by the intrusion detection community over the prior several years to detect a large number of different security problems, and is a good example of the state of the art in Snort rule development as of December 2010; how Snort was typically used, and the kinds of problems that the community was using it to detect, and in particular, the things a POSA would readily know how to do given both these rules and the Sourcefire manual.

206. As I explained above, I personally used Snort rules extensively before 2010 and was made aware of the existence of the Bleeding Snort project on multiple occasions, which later became Emerging Threats. A POSA by 2010 would have been familiar with Emerging Threats as a source where custom rules could be downloaded for loading onto devices such as Sourcefire. Centripetal's own press release in 2014 confirms that Emerging Threats was founded in 2003 and by 2014 was a world-leading provider of threat intelligence rulesets:

About Emerging Threats

Emerging Threats is a world-leading provider of commercial and open source threat intelligence. Founded in 2003 as a cyber security research community, Emerging Threats has become the de facto standard in network-based malware threat detection. The company's ETOpen Ruleset, ETPro® Ruleset, and IQRisk® Suite of threat intelligence are platform agnostic for easy integration with Suricata, SNORT®, and other network intrusion protection and detection systems. With ETPro Ruleset, organizations can achieve the highest standards of malicious threat detection with world-class support and research for extended vulnerability coverage. ETPro Ruleset is ideal for enterprises, government agencies, financial institutions, SMBs, higher education, and service providers.

For more information, please visit http://www.emergingthreats.net

EX-1082.

207. As described above in Section IX A, Sourcefire explicitly describes how to import text files of rules into the product (such as those provided by Bleeding Snort/Emerging Threats) and provides a user-interface panel for the selection and import of such files of text rules. EX-1004 pp. 1857-1859. Sourcefire sets rules to be disabled upon import, and provides instructions for users to set the rule state (e.g., alert, drop, or disabled) so that users can tune the ruleset to match their network environment. EX-1004 pp. 1858, 435-438.

208. It would have been obvious to a POSA to use this functionality provided for uploading text files of Snort rules, since the Bleeding Snort/Emerging Threats rules were distributed as text files explicitly formatted as Snort rules and

advertised for that purpose. Sourcefire was designed so that Snort rules could be imported and Emerging Threats published rules designed specifically to be imported into Snort-based systems, such as Sourcefire. Thus in my opinion, a POSA aware of the Sourcefire manual would very naturally be motivated to combine it with contemporary sets of rules, such as the Emerging Threats rules. Further, Sourefire expressly teaches users to select the rule state (e.g., alert, drop, or disabled) after importing to tune the rules to their network. Thus, in my opinion a POSA would have been motivated to set Emerging Threats rules imported onto a Sourcefire IPS to drop traffic where such a rule would be aligned with the user's network security goals (e.g., to block exfiltration of top secret information or social security numbers).

209. I have reviewed the prosecution history of the '343 patent in its entirety and can find no record of the Examiner considering the Emerging Threats rules. In my opinion, this lack has led to the Examiner erring in considering the '343 claims to be patentable. Similarly, in IPR2021-01155, Palo Alto Networks did not bring to the Board's attention any Snort rules showing that Sourcefire was very capable of detecting exfiltrations, and had actually been used for that purpose long before March 2013. Thus the Examiner and Board have not had the opportunity to consider very important aspects of the prior art.

210. I note that Patent Owner appears to have been aware of Emerging Threats rules, because they produced a press release indicating that they licensed certain rulesets in 2014 and describing them at that time:

> …Emerging Threats has become the de facto standard in network-based malware threat detection. The company's ETOpen Ruleset, ETPro® Ruleset, and IQRisk® Suite of threat intelligence are platform agnostic for easy integration with Suricata, SNORT®, and other network intrusion protection and detection systems.

EX-1082. Nonetheless, Patent Owner did not disclose any Emerging Threats rules to the Patent Office during prosecution of the '343 patent.

211. In my opinion, EX-1020 contains a large number of rules that are specifically targeted at detecting exfiltration of information from networks (as opposed to attacks or exploits on the networks).

212. As a helpful illustration of the types of data that might be exfiltrated, I quote a declaration by Sean Moore (the named inventor of the '343 patent):

> Data exfiltrations occur when malware and/or malicious actors carry out an unauthorized data transfer from a computer. For example, a cyber-criminal might steal data such as username/passwords, personal information, financial data, or corporate trade secrets, and exfiltrate that data out from a computer or network and into their possession.

EX-1081 p. 4. In the remainder of this section, I will analyze in detail four rules that a POSA would readily have understood as intended to look for exfiltration in

progress. These are just a few examples out of the many rules disclosed in Emerging Threats for detecting and protecting against exfiltrations.

213. For example, consider the following Snort Rule:

```
#US Top Secret Not Releasable to Foreign Nationals
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
$HTTP_PORTS (msg:"ET POLICY HTTP - US Top Secret
NOFORN"; flow:to_server,established;
pcre:"/(TOP\sSECRET|TS)//[\s\w,/-]*NOFORN[\s\w,/-
]*(?=//(25)?X[1-9])/ism"; classtype:policy-
violation;
reference:url,doc.emergingthreats.net/bin/view/Main
/2002526;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_Classified_Inform
ation; sid:2002526; rev:4;)
```

(EX-1020, SID 2002526). A POSA familiar with Snort Rules would understand the purpose of this rule. EX-1004, Ch. 20. Specifically, the protocol of the rule is "tcp", indicating that Snort was to inspect the contents of TCP connections. EX-1004 pp. 764, 766. The presence of the '`->`' indicates only packets in one direction of the TCP connection were to be examined for this rule. EX-1004 pp. 764, 769. Then, the keywords "`flow:to_server,established`" indicate that the direction of traffic to be inspected was from a client to a server. EX 1004, pp. 820-822. Specifically, the packets to be examined were from a client in "`$HOME_NET any`" indicating the client had an IP address on the local (protected) network, and any client port, and that the server was located

somewhere in "$EXTERNAL_NET $HTTP_PORTS", that is on an external network with a server port characteristic of HTTP. EX-1004, pp. 214-216, 372-376.

214. Thus, the rule is intended to examine traffic in requests from HTTP clients (such as web browsers) on the protected network to HTTP servers on the rest of the internet. This is notable from an exfiltration perspective as it was common practice to allow HTTP requests out through the firewall relatively freely so that employees or other uses of the networks could make use of the world wide web. EX-1001 1:31-40.

215. The rule then applies the test "pcre:"/(TOP\sSECRET|TS)//[\s\w,/-]*NOFORN[\s\w,/-]*(?=//(25)?X[1-9])/ism";". This uses Snort's pcre keyword to specify a regular expression to search for throughout the HTTP request packets (Snort adapts the popular regular expression syntax of the computer language Perl). EX-1004 pp. 800-809. Specifically, this is searching for the strings "TOP SECRET" or "TS" along with "NOFORN" separated by appropriate spaces and punctuation. Thus, this regular expression is searching for markings in documents or content flowing towards external servers indicating that the information was classified as

top secret and not for distribution to foreign (i.e. Non-US) nationals. EX-1021 pp. 5, 14.

216. A POSA would understand that this is a classic exfiltration use case. The rule is not looking for attacks on the network, but rather for classified information leaving the network in the headers or bodies of outbound HTTP requests.

217. For a second example, consider this rule:

```
#by Matt Jonkman
#
alert tcp $HOME_NET $HTTP_PORTS ->
$EXTERNAL_NET any (msg:"ET ATTACK_RESPONSE
Possible /etc/passwd via HTTP (linux style)";
flow:established,from_server;
content:"root|3a|x|3a|0|3a|0|3a|root|3a|/root|3
a|/"; nocase; classtype:misc-activity;
reference:url,doc.emergingthreats.net/bin/view/
Main/2002034;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/ATTACK_RESPONSE/ATTACK_RESP
ONSE_etc-passwd; sid:2002034; rev:8;)
```

(EX-1020, SID 2002034). The message of the rule (which would appear in the Sourcefire UI were this rule to fire) suggests its purpose: `msg:"ET ATTACK_RESPONSE Possible /etc/passwd via HTTP (linux style)"`. A POSA would know that /etc/passwd is the text file where password information is stored on Unix and Linux systems, and the rule thus suggests a

possible unauthorized transmission of the password file out of the organization via a web request (HTTP). EX-1028.

218. The details of the rule substantiate this impression. The protocol of the rule is "tcp", indicating that Snort was to inspect the contents of TCP connections. EX-1004 pp. 764, 766. The presence of the '`->`' indicates only packets in one direction of the TCP connection were to be examined for this rule. EX-1004 pp. 764, 769. Then, the keywords "`flow: established, from_server`" indicate that the direction of traffic to be inspected was from a server to a client. EX 1004, pp. 820-822. Specifically, the packets to be examined were from a server in "`$HOME_NET $HTTP_PORTS`" indicating the server had an IP address on the local (protected) network, and a server port indicative of being an HTTP server, and that the client was located somewhere in "`$EXTERNAL_NET any`", that is on an external network with any client TCP port. EX-1004, pp. 214-216, 372-376.

219. Thus, this rule was examining all traffic outbound from web servers on the protected network and going towards web clients anywhere on the Internet somewhere. To these outbound HTTP response packets, the application layer matching operator being applied is:

```
content:"root|3a|x|3a|0|3a|0|3a|root|3a|/root|3a|/"
; nocase;
```

The Snort "content" keyword matches literal strings in the packets being searched, with hexadecimal characters being encoded between '|'. Thus "`|3a|`" is a colon. EX-1029. Thus the Snort rule is looking for "`root:x:0:0:root:/root:/`" (In this case modified by the "`nocase`" to match either upper or lower case. EX-1004 pp. 778, 831-832.

220. Compare this to the first line in the example password file from exhibit EX-1028:

```
root:x:0:0:root:/root:/bin/ash
```

EX-1028. A POSA would understand that the content keyword in this Snort rule is looking for the beginning of the first line of a UNIX password file. The password file leaving the network in an HTTP response is a strong indication of the exfiltration of sensitive information from a local web server.

221. For a third example, consider the rule with SID 2001328:

```
#Submitted by Patrick Harper. pcre by Matt Jonkman
#This rule is disabled by default. It should
generally be run on the outside of your network,
not internally. Enable it where useful.
#
#alert ip any any -> any any (msg:"ET POLICY SSN
Detected in Clear Text (dashed)"; pcre:"/ ([0-
6]\d\d|7[0-256]\d|73[0-3]|77[0-2])-\d{2}-\d{4} /";
```

```
classtype:policy-violation;
reference:url,doc.emergingthreats.net/2001328;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSN_in_the_Clear
; sid:2001328; rev:13;
```

(EX-1020, SID 2001328). The message (`ET POLICY SSN Detected in Clear Text (dashed)`) suggests that the rule is looking for social security numbers in the clear (i.e. unencrypted) which on most networks is a bad sign of data being mishandled (hence the classtype here being "`policy-violation`").

222. In contrast to the two prior examples, this rule is examining all IP packets in all directions (`ip any any -> any any`). EX-1004 pp. 764, 766, 769. However, the suggestion in the comment above the rule is to run it on the "outside of your network" – i.e. outside the network firewall, where a sensor running this rule would be able to see both data leaving the network (possibly being exfiltrated), or entering the network (possibly after being exfiltrated from somewhere else).

223. The application layer test is

```
pcre:"/ ([0-6]\d\d|7[0-256]\d|73[0-3]|77[0-2])-
\d{2}-\d{4} /"
```

This regular expression is searching for three groups of digits, separated by '-' characters. The second group (`\d{2}`) means two ASCII digits, while `\d{4}` is

a group of four ASCII digits. EX-1004 pp. 802, 804. Thus, these groups match the familiar -XX-XXXX pattern at the end of social security numbers.

224. The first group "`([0-6]\d\d|7[0-256]\d|73[0-3]|77[0-2])`" matches three digits, but also includes some complex constraints on them. In the PCRE syntax, the vertical '|' characters when unescaped indicate alternatives – in this case, three possibilities are allowed. EX-1004 p. 803. The first is "`([0-6]\d\d`" which indicates any digit between 0 and 6 inclusive, followed by two more digits. This represents the common case of area assigned social security numbers. EX-1034. The other alternatives look for rarer forms of the first three digits and need not concern us here. In sum, this Snort rule is not looking for an attack, but instead is looking for the movement of social security numbers in clear text in or out of the network – i.e., it targets cases of exfiltration.

225. For a fourth example, consider this rule:

```
#Application data stream
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
```

```
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021). The message of the rule again suggests its purpose: "`msg:"ET POLICY TLS/SSL Encrypted Application Data on Unusual Port SSLv3`". A POSA would understand that this is looking for transmission of encrypted data on an unusual port.

226. The details of the rule substantiate this impression. The protocol of the rule is "tcp", indicating that Snort was to inspect the contents of TCP connections. EX-1004 pp. 764, 766. The presence of the '`->`' indicates only packets in one direction of the TCP connection were to be examined for this rule. EX-1004 pp. 764, 769. Then, the keywords "`flow: established, to_server`" indicate that the direction of traffic to be inspected was from a client to a server. EX 1004, pp. 820-822. Specifically, the packets to be examined were from a client in "`$HOME_NET any`" indicating the client had an IP address on the local (protected) network, and any client port, and that the server was located somewhere in "`$EXTERNAL_NET 1024:65535`", that is on an external network with a server port that is not in the range used for well-known system protocols. EX-1004, pp. 214-216, 372-376, EX-1027.

227. The operator being applied to packets of this profile was, in part, "`flowbits:isset,BS.SSL.Established`"

(which checks that earlier rules had established that this was an ongoing SSL connection). EX-1004 pp. 851-855. The remainder of the operator was:

```
content:"|17 03 00|"; depth:4;
```

This is looking for a particular SSL/TLS record header. The structure of that header is as follows:

```
        record type (1 byte)
       /
      /    version (1 byte major, 1 byte minor)
     /    /
    /    /         length (2 bytes)
   /    /         /
+----+----+----+----+----+
|    |    |    |    |    |
|    |    |    |    |    | TLS Record header
+----+----+----+----+----+


Record Type Values       dec      hex
-------------------------------------
CHANGE_CIPHER_SPEC        20     0x14
ALERT                     21     0x15
HANDSHAKE                 22     0x16
APPLICATION_DATA          23     0x17


Version Values            dec     hex
-------------------------------------
SSL 3.0                   3,0  0x0300
TLS 1.0                   3,1  0x0301
TLS 1.1                   3,2  0x0302
TLS 1.2                   3,3  0x0303
```

EX-1038, pp. 16-17. The Snort content operator is looking for hex bytes |17| (corresponding to APPLICATION_DATA) followed by 03 and 00 (corresponding to SSL 3.0 major and minor version numbers. The “depth 4” in the operator above indicates that the search for the content cannot go more than 4 bytes into the packet payload (i.e. it’s looking for the header at the beginning of the record). EX-1004 p. 783.

228. Thus this rule was examining all traffic outbound from the protected network and going towards a high-numbered server port somewhere on the Internet, and indicating the transmission of application data via SSL v3.0. A POSA would understand that this rule was not directed to looking for instances of exploits or attacks (which would be invisible within the encryption) but rather was looking for exfiltration of some kind of encrypted data. At Fireeye, in our web analysis product, I remember working on a similar feature where we looked for large encrypted transmissions outbound from the network on non-HTTP ports as a possible symptom of exfiltration of large amounts of data. I recognize rule 2003021 as similar in spirit to my work there.

229. A POSA would recognize that the operator in rule 2003021 is very similar to the 'Require-TLS' operator of the '343 patent. EX-1001 7:41-49, 8:7-38. That operator is described as checking.

> The application packet contained in the IP packet is a TLS Record Protocol packet, the value of the version field in the TLS Record Protocol packet header may be compared to the values that encode version numbers 1.0, 1.1, and 1.2.

230. Based on this determination, the packet could be blocked or allowed to carry on. The '343 patent considers this to be a method of exfiltration detection and prevention. A POSA would understand that the Snort rule above is also

looking for the TLS header in IP addresses, checking for a particular version, and then alerting based on its presence. A POSA would know that the rule could be trivially changed from "alert" to "drop" or "pass" to achieve the functionality described in the patent. EX-1004 pp. 358-359, 761, 765.

231. Sourcefire and Emerging Threats thus provide examples of the "operator" as disclosed and claimed in the '343 patent. The Snort rules include an operator that specifies application header field criteria matching the rule and specifies a packet transformation function to apply to packets with matching application header field values.

232. As I noted above, Patent Owner's press release in 2014 shows that not only did Patent Owner know of the Emerging Threats rules, but Patent Owner asserted that they were designed for integration with Snort-based network intrusion protection systems, like the one at the heart of Sourcefire, and that Emerging Threats was a "de facto standard" in "malware threat detection". EX-1082. Nonetheless, Patent Owner never disclosed any Emerging Threats rules to the Patent Office during prosecution of the '343 patent, and, during proceedings over both the '193 and '343 patents, repeatedly made arguments suggesting that it was unknown to use Sourcefire for exfiltration detection, despite the presence of

numerous rules obviously targeted at exfiltration as I discuss earlier in this section. EX-1083 p. 22, EX-1084 pp. 19, 31, 40-43, 47-49.

233. For example, in its response to Palo Alto's Petition for IPR2021-01155 (of the '343 patent) Patent Owner stated:

> Here, Petitioner's obviousness ground attempting to fill the exfiltration gap is legally insufficient as it completely fails to allege a motivation to modify *Sourcefire* to prevent exfiltration or a reasonable expectation of expectation of success in doing so.

EX-1084 p. 58. On the contrary, the Emerging Threats rules were indeed a widely known "de-facto standard" (as avowed by Patent Owner in EX-1082) and explicitly and publicly disclose numerous rules directly targeting exfiltration use cases, thus leaving a POSA motivated to block exfiltrations using these Snort rules, or obvious variations of them, with Snort-based systems, such as Sourcefire, and reasonably expecting success.

234. Similarly in response to the IPR petition for 2018-01559 (of the '193 IPR), Patent Owner stated:

> As Petitioner explicitly notes, the '193 Patent is capable of preventing "a category of cyber-attacks known as exfiltrations"…On the other hand, the Sourcefire reference is the user manual for an "Intrusion Prevention System," or "IPS"…The entirety of the 2123-page Sourcefire reference fails to mention the term exfiltration and

> explicitly touts itself as "provid[ing] best-in-class ***intrusion*** detection and prevention."

EX-1083 pp. 25. However, Emerging Threats shows that Snort-based systems, such as Sourcefire, previously were used to prevent exfiltrations.

E. Detailed Invalidity Analysis

235. As detailed below, in my opinion, Sourcefire, and the associated Emerging Threats Rules, render obvious claims 1-20 of the '343 Patent. Each of the limitations of these claims are disclosed by or are obvious in view of the prior art and any potential differences that may be argued by the Patent Owner are implementation details that are well-within the knowledge and capability of a POSA. Because claims 1, 8, and 15, and the respective dependent claims, largely mirror each other, my analysis below groups counterpart claims together.

1. Independent Claims 1, 8, and 15

[1pre] A method comprising: [receiving, by a computing system comprising memory and at least one processor,][7]

[8pre] An apparatus comprising: at least one processor; and memory storing instructions that, when executed by the at least one processor, cause the apparatus to:

[7] I include part of element 1a here because its recitation of a computing system comprising memory and at least one processor overlaps with the preambles of claims 8 and 15.

[15pre] One or more non-transitory computer-readable media comprising instructions that, when executed by one or more computing devices, cause the one or more computing devices to:

236. Sourcefire discloses packet-filtering methods. EX-1004 p. 222, 226-227. Sourcefire describes that 3D Sensors with Intrusion Prevention System (IPS) monitored a network for attacks that might affect the availability, integrity, or confidentiality of hosts on the network. Each deployed 3D Sensor with IPS (*computing system, apparatus,* or *computing device*) received and examined the packets that traversed a network, looked for malicious activity, and reported and/or took actions based on detected malicious activity.

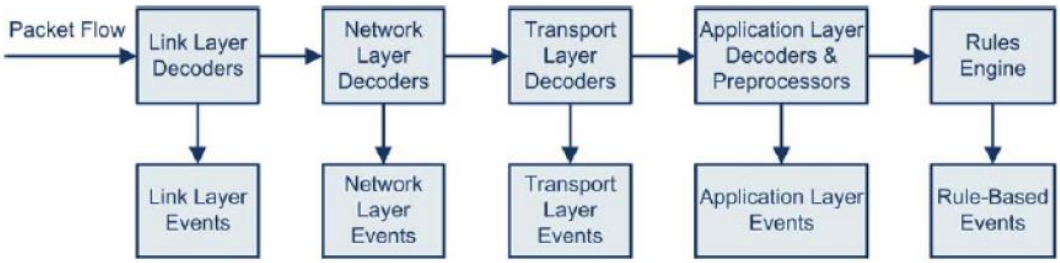
237. Claim 1 recites that the computing system comprises "memory and at least one processor". Claim 8's preamble calls for the apparatus to have "at least one processor" and "memory comprising instructions that, when executed by the at least one processor, cause the apparatus to." Claim 15's preamble recites "[o]ne or more non-transitory computer-readable media comprising instructions that, when executed by one or more computing devices, cause the one or more computing devices to". Sourcefire discloses all three of these. Sourcefire discloses that each 3D Sensor with IPS included a processor and memory that executed instructions to perform tasks, such as dropping or allowing packets according to rules employed by the 3D Sensor. EX-1004 pp. 222-223, 234-235, 253-254, 257, 262-264, 271, 435-439.

238. Specifically, Sourcefire discloses a statistic that "Displays the current CPU usage statistics appropriate for the platform for all CPUs on the sensor." EX-1004 p. 2050. Thus a POSA would understand that the appliance had a processor (CPU). Similarly, Sourcefire discloses monitoring of the total memory, memory in use, and available memory. EX-1004 p. 2056. A POSA further understood that each 3D Sensor with IPS included software (i.e., instructions) on a disk (i.e., a non-transitory computer-readable medium) that would cause the 3D Sensor to perform the tasks set forth in the claim elements. For example, Sourcefire discloses a "Disk Usage Widget" which "indicated the percentage of space used on each partition of the appliance's hard drive. It also shows the capacity of each partition." Ex-1004 p. 82. Thus the POSA would infer the existence of a disk on which the system and application software for the platform could be stored.

[1a] receiving, by a computing system comprising memory and at least one processor, a plurality of packets wherein the plurality of packets comprises a first portion of packets and a second portion of packets;

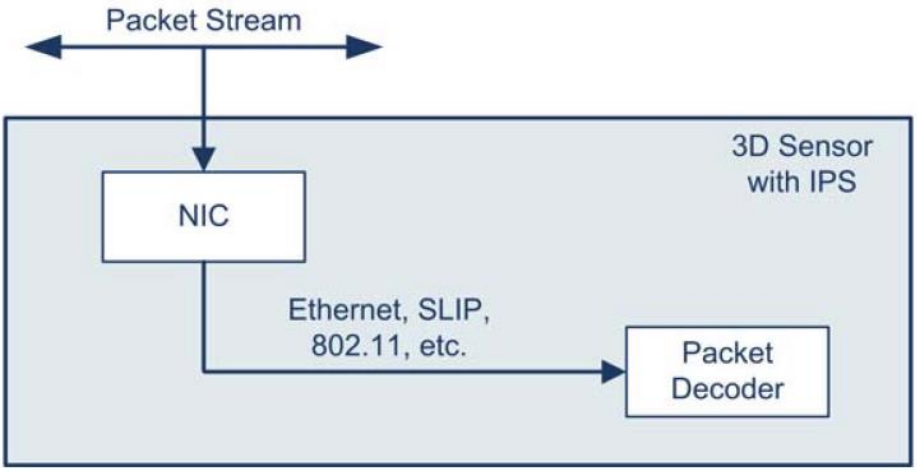
[8a], [15a] receive a plurality of packets;

239. Sourcefire discloses packet-filtering methods. EX-1004 p. 222, 226-227. The IPS performed these functions on received packets using a plurality of packet-filtering rules implemented through a series of decoders, preprocessors, and a rules engine (as illustrated below), each of which implemented rules to look for the broad range of exploits that attackers had developed:



EX-1004 pp. 34-35, 254-256.

240. Sourcefire discloses that each 3D Sensor received packets through a communication interface. EX-1004 p. 222. Two of these communication interfaces were "inline" interfaces in which decoder rules, preprocessor rules, and intrusion rules dropped or allowed packets received into such decoders, preprocessors, and the rules engine via the inline interface of the 3D Sensor. Sourcefire discloses one example of how the 3D Sensor received packets via a network interface card as shown below:



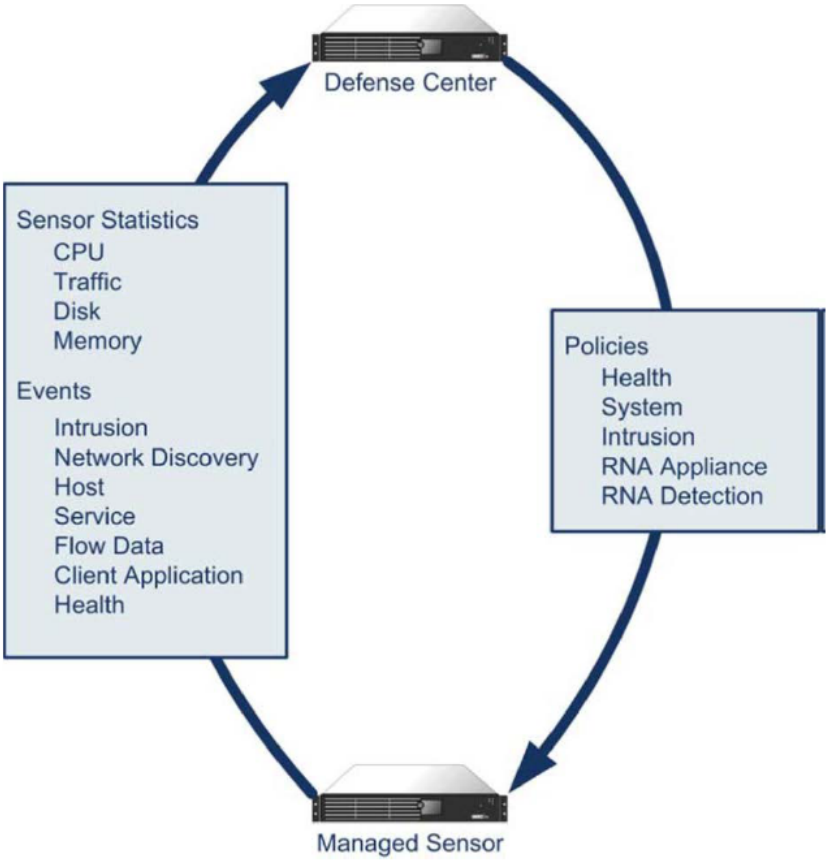
EX-1004 pp. 222-223, 234-235, 253-254, 257, 262-264, 435-439.

241. Thus, Sourcefire discloses 3D Sensors receiving a plurality of packets, any subset of which can be deemed to be a first portion of packets and any other subset of which can be deemed a second portion of packets. EX-1004 pp. 213-214, 225-227, 234-235, 260, 277-278, 333-335, 373-379.

[1b] determining, based on a packet header field value, whether each packet of the plurality of packets comprises data corresponding to first criterion specified by one or more packet-filtering rules;

[8b], [15b] determine, based on a packet header field value, whether the plurality of packets comprises data corresponding to first criterion specified by one or more packet-filtering rules;

242. Sourcefire's 3D Sensors received packet-filtering rules from the Defense Center, the rules corresponding to customizable intrusion policies of the user, and as a result of importing an intrusion policy, Security Enhancement Update, or local rules files to the 3D Sensor. EX-1004 pp. 34, 88, 105-107, 338-350, 1848-1859, 2001-2002, 2083-2084, 2093.



> You can . . . push policies created on the Defense Center and software updates to managed sensors. If you have software sensors or Intrusion Agents on your network, you must use the Defense Center to manage them.

EX-1004 p. 34.

> You can create [a system] policy on the Defense Center and push it to the appropriate sensors instead of replicating it locally.

EX-1004 p. 105.

243. Sourcefire's 3D Sensors used these received packet-filtering rules to identify packets including data (*e.g.*, 5-tuple) corresponding to characteristics associated with malicious activities. EX-1004 p. 34.

> By placing 3D Sensors with IPS on key network segments, you can examine the packets that traverse your network for malicious activity. Each 3D Sensor with IPS uses rules, decoders, and preprocessors to look for the broad range of exploits that attackers have developed.

EX-1004 p. 34.

244. The '343 patent gives examples of the "packet header field values" in Figure 3 as IP protocol values, ranges of source IP addresses, destination IP addresses, source ports, and destination ports. EX-1001 Fig. 3, 5:50-6:11. This data is known in the relevant art as "5-tuple" data. For example:

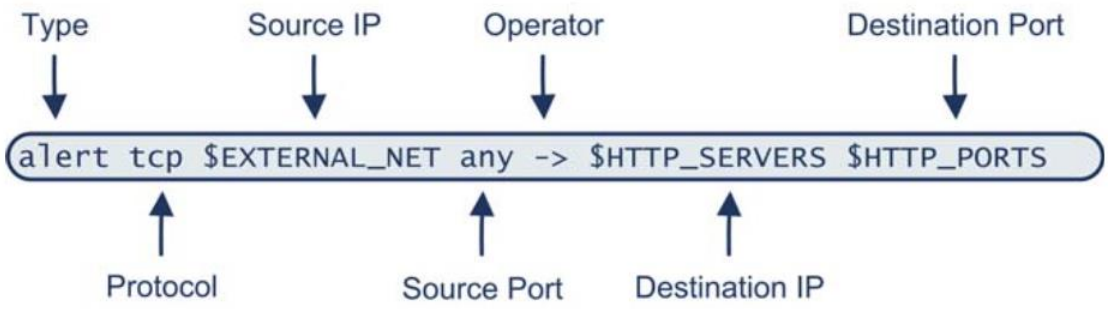
> The filtering process described herein may be viewed as having two (2) stages: A first stage in which the "5-tuple" of IP packet header field values and transport protocol (e.g., TCP, UDP, etc.) packet header field values may be filtered;…

EX-1001 8:39-42.

245. Like the '343 patent, Sourcefire describes that all of the intrusion rules had 5-tuple criteria associated with them. EX-1004 pp. 256-259, 761-770. The 5-tuple criteria included the source IP addresses and port and the destination IP addresses and port. EX-1004 pp. 358, 488-490, 761-762. Sourcefire describes that

the IP address 5-tuple rule criteria could be a range of IP addresses. EX-1004 pp. 762-763, 767, 769, 702-704, 59-60. For example:

> Every standard text rule and shared object rule has parameters and arguments that make up the rule header. For example, the following illustrates the parts of a rule header:



EX-1004 p. 764.

246. As in the '343 patent, the rule header was evaluated first, before any consideration of application layer data (thus corresponding to the 1b determining step I am considering here). EX-1004 p. 764; EX-1018 p. 4.

247. Thus, 3D Sensors received a number of packet-filtering rules that included criteria (the rule header) configured to cause the Sensor to determine that certain packets corresponded to one or more packet-header field values specified in those criteria. To meet the claim element 1b, a packet-header field value present in every packet (such as the IP addresses or IP protocol, which are present even when packets are subject to IP fragmentation) could be the "packet header field value" of the first criterion. A packet header field value such as a TCP port that might only

be present in some packets (for example if packets were fragmented) could alternatively be the "packet header field value." The claim elements 8b or 15b similarly call for determining whether the packet header field is data comprised in the plurality of packets. In my opinion, this would also read on a case where the first criterion selected a packet header field value such as an IP address, protocol, or TCP port.

248. In any case, since Sourcefire's rule headers can decide based on any value in the 5-tuple, in view of the knowledge, skill and creativity of a POSA, Sourcefire discloses this limitation.

[1c] responsive to a determination by the computing system that a packet header field value of the first portion of the packets comprises data corresponding to the first criterion specified by at least one matching packet-filtering rule, applying, by the computing system and to each packet in the first portion of packets, one or more operators specified by the at least one matching packet-filtering rule;

[8c], [15c] responsive to a determination that a packet header field value of a first portion of packets comprises data corresponding to the first criterion specified by at least one matching packet-filtering rule, apply, to each packet in the first portion of packets, one or more operators specified by the at least one matching packet-filtering rule;

249. The portion of Claim element 1c prior to "applying", simply reiterates the Claim element 1b and indicates that the remainder of element 1c occurs "responsive" to a determination. See Claim 1b above. Similarly, the portion of Claim element 8c and 15c prior to "apply" reiterate claim element 8b/15 b and indicate that the remainder of those claims occurs "responsive" to a determination,

except applied to a subset of the plurality of packets – the "first portion" which could be any subset of the packets. See Claim elements 8b/15b above.

250. The new text in Claim 1c/8c/15c says that when the first criterion of 1b/8b/15b has been met, then "one or more" operators specified by the "one or more" packet filtering rules, are to be applied to every packet in the first portion of packets. As discussed above (see Section VIII A), the '343 patent "operator" is to be interpreted as some text in a rule that will invoke the functionality of the source code of the system. It may determine application header field values in packets seen by the system, and either allow the packets or drop them according to those values. I further discuss in Section IX C above which keywords in the Snort/Sourcefire rule options can be used as an operator (or part of an operator) in this sense.

251. Because the rule option keywords in Snort are not checked at all unless the five-tuple fields are matched (Section IX C), Sourcefire discloses the limitation of invoking the "operator" "responsive to the determination" that the packet header field value specified in the "first criterion" has been met. Thus, Sourcefire discloses or renders obvious element 1c.

[1d] determining, based on an application header field value, the second portion of packets based on whether the first portion of packets comprises data corresponding to second criterion specified by one or more operators specified by the at least one matching packet-filtering rule; and

[8d], [15d] determine, based on an application header field value, a second portion of packets based on whether the first portion of packets comprises data corresponding to second criterion specified by one or more operators specified by the at least one matching packet-filtering rule; and

252. Sourcefire discloses that the "rule options" portion of the rule (second criterion) could be used to detect particular application header field values. I discuss at length in IX B above which portions of the Snort rule language could be used to specify the '343 patent's "operator".

253. In particular, Sourcefire discloses (in a section titled "HTTP Content Options") that rule options can look for particular values of the HTTP method, the HTTP request URI, the HTTP Cookie Header, etc. All of these are examples of "application header field values". EX-1004 pp. 783-788. Specifically, Sourcefire discloses that rule options can assess

> the request method field, which identifies the action such as GET, PUT, CONNECT, and so on to take on the resource identified in the URI.

EX-1004 p.786.

254. This example is also one of the few explicitly disclosed in the '343 patent – specifically, the HTTP-EXFIL operator is defined to make a decision whether to block or allow packets solely based on the HTTP method in an HTTP request:

> If the application packet contained in the IP packet is an HTTP packet, the value of the HTTP method field in the HTTP packet header may be compared to the values that encode the GET, POST, PUT, DELETE, and CONNECT methods. If a match is found, then the HTTP-EXFIL operator may return either ALLOW or BLOCK, depending on the method value.

EX-1001 7:65-8:4.

255. Here, the "second portion of packets" would be those in connections that had the required value for the application header field values.

256. For a second example, as I discussed above, a POSA understood that Sourcefire describes the use of SSL/TLS rule keywords to invoke the application layer SSL preprocessor and extract information about SSL or TLS version and session state from SSL/TLS record headers in packets for an encrypted session. EX-1004 pp. 825-828, 491, 597-601, 697-700. A POSA understood that SSL was a widely adopted standard technology[8] for keeping an internet connection secure and safeguarding any sensitive data that is being sent between two systems. A POSA also understood that TLS was widely adopted as an updated, more secure, version of SSL. Sourcefire discloses that the SSL preprocessor (including when

[8] A POSA understood SSL version 2.0 was published in 1995 and TLS version 1.0 was published as a newer version of SSL in 1999.

invoked by the rules engine) implemented the SSL preprocessor rules and intrusion rules including SSL keywords (*e.g.*, SSL_version) for both SSL and TLS. EX-1004 pp. 827-828, 491, 596-601, 700.

257. SSL versions are application header field values, as evidenced by the example in the patent of the REQUIRE-TLS-1.1-1.2 operator which picks the version number out of an SSL connection and makes a determination which packet transformation function to use on the packet (BLOCK or ALLOW). EX 1001 8:20-38.

258. Consistent with my analysis, I understand that the appeals court affirmed the Board's finding in the '552 IPR that Sourcefire disclosed determining whether application-header-field-values correspond to a TLS version. EX-1069 at pp. 17-18.

259. Not only does Sourcefire disclose the possible use of rule options inspecting application header field values, we also have the evidence of the Emerging Threats rules that the open source intrusion detection community did in fact write and disseminate rules doing this long before March 2013. For example, consider the example discussed at length above in Section IX C of the rule with SID 2003021:

```
#Application data stream
#
```

```
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021). The message of the rule again suggests its purpose: "`msg:"ET POLICY TLS/SSL Encrypted Application Data on Unusual Port SSLv3`". A POSA would understand that this is looking for transmission of encrypted data on an unusual port.

260. The operator being applied to packets of this profile was, in part,

"`flowbits:isset,BS.SSL.Established`"

(which checks that earlier rules had established that this was an ongoing SSL connection). EX-1004 pp. 851-855. I note this still satisfies claim 1, which allows for the first and second criterion coming from "one or more rules". The remainder of the operator was:

```
content:"|17 03 00|"; depth:4;
```

This is looking for a particular SSL/TLS record header. The structure of that header is as follows:

```
        record type (1 byte)
       /
      /    version (1 byte major, 1 byte minor)
     /    /
    /    /         length (2 bytes)
   /    /         /
+----+----+----+----+----+
|    |    |    |    |    |
|    |    |    |    |    | TLS Record header
+----+----+----+----+----+
```

Record Type Values	dec	hex
CHANGE_CIPHER_SPEC	20	0x14
ALERT	21	0x15
HANDSHAKE	22	0x16
APPLICATION_DATA	23	0x17

Version Values	dec	hex
SSL 3.0	3,0	0x0300
TLS 1.0	3,1	0x0301
TLS 1.1	3,2	0x0302
TLS 1.2	3,3	0x0303

EX-1038, pp. 16-17. The Snort content operator is looking for hex bytes |17| (corresponding to APPLICATION_DATA) followed by 03 and 00 (corresponding to SSL 3.0 major and minor version numbers. The "depth 4" in the operator above indicates that the search for the content cannot go more than 4 bytes into the packet payload (i.e., it's looking for the header at the beginning of the record). EX-1004 p. 783.

261. A POSA would recognize that this operator is very similar to the 'Require-TLS' operator of the '343 patent. EX-1001 7:41-49, 8:7-38. That operator is described as checking:

> If the application packet contained in the IP packet is a TLS Record Protocol packet, the value of the version field in the TLS Record

> Protocol packet header may be compared to the values that encode version numbers 1.0, 1.1, and 1.2.

EX-1001 8:30-34. Again, the second portion of packets would be those matching the application header field value in the operator(s). Thus in my opinion, Sourcefire and the Emerging Threats rules both disclose this limitation.

[1e] responsive to determining the second portion of packets that comprises data corresponding to the second criterion specified by one or more operators specified by the at least one matching packet-filtering rule, applying, by the computing system and to each packet in the second portion of packets that match the second criterion, at least one packet transformation function configured to prevent an exfiltration operation, wherein the at least one packet transformation function indicates whether each packet in the second portion of packets is allowed to continue toward its destination.

[8e] responsive to determining the second portion of packets that comprises data corresponding to the second criterion specified by one or more operators specified by the at least one matching packet-filtering rule, apply, to each packet in the second portion of packets, at least one packet transformation function configured to prevent an exfiltration operation, wherein the at least one packet transformation function indicates whether each packet in the second portion of packets is allowed to continue toward its destination.

[15e] responsive to determining the second portion of packets that comprises data corresponding to the second criterion specified by one or more operators specified by the at least one matching packet-filtering rule, apply, to each packet in the second portion of packets that match the second criterion, at least one packet transformation function configured to prevent an exfiltration operation, wherein the at least one packet transformation function indicates whether each packet in the second portion of packets is allowed to continue toward its destination.

262. Sourcefire discloses that each rule has a rule action that can be either "alert", "pass" or "drop". Pass and alert rules allow the packet(s) through, while drop rules will block them. Thus the Sourcefire rules specify a packet

transformation function that will "indicate whether each packet in the second portion of packets is allowed to continue toward its destination.".

263. Further, as discussed at length in Section IX D above, the Emerging Threats rules contain a number of instances of rules specifically targeted at exfiltration. To continue with the example we have been using throughout this Claim 1/8/15 discussion, we return to SID 2003021:

```
#Application data stream
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021).

264. The message of the rule suggests its purpose: "`msg:"ET POLICY TLS/SSL Encrypted Application Data on Unusual Port SSLv3`". A POSA would understand that this is looking for transmission of encrypted data on an unusual port via SSLv3 (a very out-of-date version of SSL, even in 2013).

265. The details of the rule substantiate this conclusion. The protocol of the rule is "tcp", indicating that Snort was to inspect the contents of TCP connections. EX-1004 pp. 764, 766. The presence of the '`->`' indicates only packets in one direction of the TCP connection were to be examined for this rule. EX-1004 pp. 764, 769. Then, the keywords "`flow: established, to_server`" indicate that the direction of traffic to be inspected was from a client to a server. EX 1004, pp. 820-822. Specifically, the packets to be examined were from a client in "`$HOME_NET any`" indicating the client had an IP address on the local (protected) network, and any client port, and that the server was located somewhere in "`$EXTERNAL_NET 1024:65535`", that is on an external network with a server port that is not in the range used for well-known system protocols. EX-1004, pp. 214-216, 372-376, EX-1027.

266. The operator second criterion being applied to packets of this profile was, in part,

"`flowbits:isset,BS.SSL.Established`"

(which checks that earlier rules had established that this was an ongoing SSL connection). EX-1004 pp. 851-855. The remainder of the operator second criterion was:

```
content:"|17 03 00|"; depth:4;
```

This is looking for a particular SSL/TLS record header. The structure of that header is as follows:

```
        record type (1 byte)
       /
      /    version (1 byte major, 1 byte minor)
     /    /
    /    /         length (2 bytes)
   /    /         /
+----+----+----+----+----+
|    |    |    |    |    |
|    |    |    |    |    | TLS Record header
+----+----+----+----+----+
```

Record Type Values	dec	hex
CHANGE_CIPHER_SPEC	20	0x14
ALERT	21	0x15
HANDSHAKE	22	0x16
APPLICATION_DATA	23	0x17

Version Values	dec	hex
SSL 3.0	3,0	0x0300
TLS 1.0	3,1	0x0301
TLS 1.1	3,2	0x0302
TLS 1.2	3,3	0x0303

EX-1038, pp. 16-17. The Snort content operator is looking for hex bytes |17| (corresponding to APPLICATION_DATA) followed by 03 and 00 (corresponding to SSL 3.0 major and minor version numbers). The "depth 4" in the operator above indicates that the search for the content cannot go more than 4 bytes into the packet payload (ie it's looking for the header at the beginning of the record). EX-1004 p. 783.

267. Thus this rule was examining all traffic outbound from the protected network and going towards a high-numbered server port somewhere on the Internet, and indicating the transmission of application data via SSL v3.0. Thus

this rule is hoping to find exfiltration of some kind of encrypted data, or prevent loss of data via the use of outdated insecure versions of SSL/TLS. See Section IX C above.

268. A POSA would understand that while Emerging Threats ships all rules as "alert" format for initial evaluation on a user network, the POSA would be motivated to change one or more rules to "drop" when tuning the rules to their network environment. As described above in Section IX A, a user of a Sourcefire system would import text rules and have them initially disabled, and would then tune their rule set to match the specific network environment, including selecting rules to activate and selecting whether to set each rule to block or alert. EX-1004 at 1857-1859, 438. For example, when using the Emerging Threats rules with Sourcefire, a user would enable select rules with an "alert" action to evaluate their performance in the user's specific network environment, and then set the rules to "drop" once the user observes that they operate as desired. This would have the effect of blocking suspected exfiltration operations of this particular kind. Meanwhile, packets in connections that match the same 5-tuple, but that did not show this pattern (of SSL v3 application field headers), would be allowed through.

269. Thus, a POSA would understand that the combination of Sourcefire and the Emerging Threats Rules would fall under claim element 1e.

2. Dependent Claims 2, 9, and 16

[2] The method of claim 1, wherein: the first portion of packets comprises a packet header field value indicating a protocol type associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

[9] The apparatus of claim 8, wherein the first portion of packets comprises a packet header field value indicating a protocol type associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

[16] The one or more non-transitory computer-readable media of claim 15, wherein: the first portion of packets comprises a packet header field value indicating a protocol type associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

270. Sourcefire discloses that the IP protocol field is part of the 5-tuple specification in the header of each rule. "In each rule header, you must specify the protocol of the traffic the rule inspects". EX-1004 p. 766. The allowed values for the field were ICMP, IP, TCP, and UDP. Thus this could be the "packet header field value" of the "first criterion" in the rule.

271. In particular, since it is required, all Emerging Threats rules have a protocol specified in their rule header. For example, consider the rule examined above in Claim element 1d:

```
#Application data stream
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
```

```
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021). Here, the "tcp" after "alert" indicates that only packets with protocol type "tcp" are to be considered by this rule, which would thus be the packet-header field value of the first criterion.

272. Thus, Sourcefire and the Emerging Threats rules disclose or render obvious these claims.

3. Dependent Claims 3, 10, and 17

[3] The method of claim 1, wherein: the first portion of packets comprises data indicating a destination port number associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

[10] The apparatus of claim 8, wherein: the first portion of packets comprises data indicating a destination port number associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

[17] The one or more non-transitory computer-readable media of claim 15, wherein: the first portion of packets comprises data indicating a destination port number associated with a particular type of data transfer, and corresponding to the first criterion specified by the at least one matching packet-filtering rule.

273. Sourcefire discloses that the destination port is part of the 5-tuple specification in the header of each rule.

> Rule headers specify the source and destination ports of the traffic the rule inspects. You can restrict packet inspection to packets originating from or destined to specific ports. This reduces the amount of packet inspection the system must perform. This also reduces false positives, making the rule more specific and removing the possibility for the rule to trigger against packets whose source and destination ports do not indicate suspicious behavior.

EX-1004 pp. 768-769. Thus, the destination port would be the "packet header field value" of the "first criterion" in the rule.

274. In particular Emerging Threats rules have a destination specified in their rule header (even if it's explicitly a wild-card). For example, consider the rule examined above in Claim element 1d:

```
#Application data stream
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021). Here, the "`1024:65535`" indicates that only packets in a connection to a server with ports between 1024 and 65535 would have the

remainder of the rule applied to them. This would thus be the packet-header field value of the first criterion.

275. Thus, Sourcefire and the Emerging Threats rules disclose or render obvious these claims.

4. Dependent Claims 4, 11, and 18

[4pre] The method of claim 1, wherein

[11pre] The apparatus of claim 8, wherein:

[18pre] The one or more non-transitory computer-readable media of claim 15, wherein

276. See Claim 1, Claim 8, and Claim 15 above.

[4a/11a/18a] the application header field value identifies one or more hypertext transfer protocol (HTTP) methods,

277. Sourcefire, in combination with the Emerging Threats rules, renders these claims obvious. Consider the rule with SID 2005319:

```
#from spyware LP data, by matt jonkman
#
alert tcp $HOME_NET any -> $EXTERNAL_NET
$HTTP_PORTS (msg:"ET MALWARE Bizconcept.info Spyware
Checkin"; flow:to_server,established; content:"POST "; depth:5;
nocase; uricontent:"/zuzu.php?&r="; nocase; classtype:trojan-
activity;
reference:url,doc.emergingthreats.net/bin/view/Main/2005319;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/MALWARE/MALWARE_Bizconcept.info;
sid:2005319; rev:3;)
```

EX-1020 SID 2005319. This rule looks for a 5-tuple consistent with requests to remote HTTP servers ("`alert tcp $HOME_NET any ->`

`$EXTERNAL_NET $HTTP_PORTS`"). If this part of the rule matched, then an operator 'content: "POST "; depth 5' looked at the beginning of the request for the HTTP request method to be a POST. POST is the HTTP method. Additionally, it checks that the request is to a particular URL "/zuzu.php?&r=". As described above in Section IX A, a user of a Sourcefire system would import text rules and have them initially disabled, and would then tune their rule set to match the specific network environment, including selecting rules to activate and selecting whether to set each rule to block or alert. EX-1004 at 1857-1859, 438. For example, when using the Emerging Threats rules with Sourcefire, a user would enable select rules with an "alert" action to evaluate their performance in the user's specific network environment, and then set the rules to "drop" once the user observes that they operate as desired.

[4b] wherein the second portion of packets comprises a plurality of HTTP packets, wherein the second portion of packets comprises a third portion of packets and a fourth portion of packets, and wherein each packet of the third portion of packets comprises at least one GET method value, and each packet of the fourth portion of packets comprises at least one of a PUT method value, a POST method value, or a CONNECT method value, the method further comprising:

[11b] wherein the second portion of packets comprises a plurality of HTTP packets, wherein the second portion of packets comprises a third portion of packets and a fourth portion of packets, and wherein each packet of the third portion of packets comprises at least one GET method value, and each packet of the fourth portion of packets comprises at least one of a PUT method value, a POST method value, or a CONNECT method value, and wherein the instructions, when executed by the at least one processor cause the apparatus to:

[18b] wherein the second portion of packets comprises a plurality of HTTP packets, wherein the second portion of packets comprises a third portion of packets and a fourth portion of packets, and wherein each packet of the third portion of packets comprises at least one GET method value, and each packet of the fourth portion of packets comprises at least one of a PUT method value, a POST method value, or a CONNECT method value, and wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to:

278. Continuing with the discussion above, if the method was GET, the operator in rule 2005319 would not match. Any packets not matching the rule would be allowed through, because as discussed above, the rule would only block packets to this 5-tuple and URL that matched the POST HTTP method.

[4c/11c/18c] allowing each packet of the third portion of packets to continue toward its respective destination; and

279. See Claim element 4b/11b/18b above. Packets in "GET" requests can be considered to be the third portion of packets and would be allowed by the rule to continue toward their respective destinations.

[4d/11d/18d] blocking each packet of the fourth portion of packets from continuing toward its respective destination.

280. See Claim element 4b/11b/18b above. Packets in "POST" requests can be considered to be the fourth portion of packets and would be dropped, and therefore blocked from continuing toward their respective destinations.

5. Dependent Claims 5 and 12

[5 pre] The method of claim 1, the method further comprising:

[12 pre] The apparatus of claim 8, wherein the instructions, when executed by the one or more computing devices, cause the apparatus to:

281. See Claims 1 and 8 above.

[5a] allowing, based on a determination that the application header field value indicates a request for specified resource data, each packet of the second portion of packets indicating a request for specified resource data to continue toward its respective destination; and

[12a] allow, based on a determination that the application header field value indicates a request for data from a specified resource, each packet of the second portion of packets indicating a request from a specified resource to continue toward its respective destination; and:

282. Sourcefire, in combination with Emerging Threats, renders this element obvious. Consider the following rule from the Emerging Threats rules:

```
#by anon 4
#
alert tcp $HOME_NET any -> $EXTERNAL_NET
$HTTP_PORTS (msg:"ET CURRENT_EVENTS
FakeAlert/FraudPack/FakeAV/Guzz/Dload/Vobfus/ZPack HTTP
Post 1"; flow:established,to_server; content:"POST "; depth:5;
uricontent:"/senm.php?data="; nocase; content:"data=";
nocase; classtype:trojan-activity;
reference:url,threatinfo.trendmicro.com/vinfo/virusencyclo/defau
lt5.asp?VName=TROJ_AGENT.GUZZ&VSect=T;
reference:url,www.threatexpert.com/report.aspx?md5=7ca709f
154e6abc678fbc4df8a3256b6;
reference:url,www.threatexpert.com/threats/trojan-fraudpack-
sd6.html; reference:url,doc.emergingthreats.net/2010234;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/CURRENT_EVENTS/CURRENT_fakealert
s; sid:2010234; rev:4;)
```

(EX-1020, SID 2010234). This rule, in the first stage, looks for connections outbound from the local network to external networks on HTTP ports (e.g. port 80). Then in the operator, it checks for ‘content:"POST "’ within the first 5 bytes

of the request. This is examining the HTTP method, and the rule will fire if the method is "POST".

283. The operator is also checking whether the request is to a specific resource on the target server, specifically it checks that 'uricontent:"/senm.php?data="', such that the second criterion will only be matched if the request is going to the url "/senm.php?data=". A POSA would understand that this rule is an exfiltration rule because, like the example provided in the '343 Patent, it is designed to detect and block HTTP POST methods. EX-1001 7:53-55.

284. As I discussed above in Section IX A, a POSA would understand that, as part of importing the Emerging Threats rules into Sourcefire and tuning their ruleset, this rule could be made a "drop" rule (instead of alert) with the desirable effect of blocking the activities of this particular Fake AV malware ("Guzz/Dload/Vobfus/ZPack"). In the drop version of the rule, when the "application header field value indicates a request for specified resource data", for example because the request method was a "GET" the rule would not fire, and the packets would be allowed through to continue toward their respective destinations. However, in the case of claim element 5b:

[5b] blocking, based on a determination that the application header field value indicates a request to submit data to be processed by a specified resource, each

packet of the second portion of packets indicating a request to submit data to be processed by a specified resource from continuing toward its respective destination.

[12b] block, based on a determination that the application header field value indicates a request to submit data to be processed by a specified resource, each packet of the second portion of packets indicating a request to submit data to be processed by a specified [resource][9] from continuing toward its respective destination; and:

285. If the request being matched against the operator specified by rule 2010234 had a request method of POST, then this would match the rule. In my opinion, it would be obvious to a POSA that an HTTP POST request, would be a "request to submit data to be processed by a specified resource". In this case, the specified resource being the URL "/senm.php?data=".

286. In the block version of this rule, the packets of the POST request would be blocked, and thus the conditions of Claims 5 and 12 would both be met. Thus, Sourcefire and the Emerging Threats rules disclose or render obvious these claims.

6. Dependent Claims 6, 13, and 19

[6] The method of claim 1, further comprising dropping packets associated with a particular type of data transfer responsive to a determination that the packets

[9] A word is missing from element 12b. For purposes of my analysis for this IPR, I assume that the word "resource" was omitted, making claim element 12b match 5b.

associated with the particular type of data transfer comprise data corresponding to a particular transport layer security (TLS) version value.

[13] The apparatus of claim 8, wherein the instructions, when executed by the one or more computing devices, cause the apparatus to drop packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of data transfer comprise data corresponding to a particular transport layer security (TLS) version value.

[19] The one or more non-transitory computer-readable media of claim 15, further comprising dropping packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of data transfer comprises data corresponding to a particular transport layer security (TLS) version value.

287. Sourcefire, in combination with Emerging Threats, renders this element obvious. The example Emerging Threats rule I used throughout the Claim 1/8/15 discussion also discloses claims 6, 13, and 19. Recall that rule was:

```
#Application data stream
#
#alert tcp $HOME_NET any -> $EXTERNAL_NET
1024:65535 (msg:"ET POLICY TLS/SSL Encrypted
Application Data on Unusual Port SSLv3";
flowbits:isset,BS.SSL.Established;
flow:established,to_server; content:"|17 03 00|";
depth:4; threshold:type limit, count 1, seconds
120, track by_src; classtype:unusual-client-port-
connection;
reference:url,doc.emergingthreats.net/2003021;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/POLICY/POLICY_SSL_TLS_on_High_P
ort; sid:2003021; rev:8;)
```

(EX-1020, SID 2003021). This is looking for a particular SSL/TLS record header. The structure of that header is as follows:

```
        record type (1 byte)
       /
      /    version (1 byte major, 1 byte minor)
     /    /
    /    /         length (2 bytes)
   /    /         /
+----+----+----+----+----+
|    |    |    |    |    |
|    |    |    |    |    | TLS Record header
+----+----+----+----+----+
```

Record Type Values	dec	hex
CHANGE_CIPHER_SPEC	20	0x14
ALERT	21	0x15
HANDSHAKE	22	0x16
APPLICATION_DATA	23	0x17

Version Values	dec	hex
SSL 3.0	3,0	0x0300
TLS 1.0	3,1	0x0301
TLS 1.1	3,2	0x0302
TLS 1.2	3,3	0x0303

EX-1038, pp. 16-17. The Snort content operator is looking for hex bytes |17| (corresponding to APPLICATION_DATA) followed by 03 and 00 (corresponding to SSL 3.0 major and minor version numbers). The "depth 4" in the operator above indicates that the search for the content cannot go more than 4 bytes into the packet payload (i.e. it's looking for the header at the beginning of the record). EX-1004 p. 783.

288. Thus, for this rule, once the first criterion was matched, the operator checked for a particular TLS/SSL version, as required by Claim 6/13/19. I note that while the claims recite data corresponding to a particular transport layer security (TLS) version, a POSA would understand that SSL 3.0 is a prior version

of the SSL/TLS protocol before TLS 1.0, as shown in SSL/TLS header structure above. EX-1038, pp. 16-17.

289. As I discussed above in Section IX A, a POSA would understand that part of importing the Emerging Threats rules into Sourcefire and tuning their packet-filtering device was deciding which rules to change to a "drop." A POSA would therefore understand that this rule could be set to drop which would have the obvious desirable effect of blocking transfers via SSL v3 on high numbered ports that might well be exfiltrations. See also Claim 1e above, as well as Section IX D.

7. Dependent Claims 7, 14, and 20

[7] The method of claim 1, further comprising dropping packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of data transfer comprise data corresponding to a particular real-time transport protocol value.

[14] The apparatus of claim 8, wherein the instructions, when executed by the one or more computing devices, cause the apparatus to drop packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of data transfer comprise data corresponding to a particular real-time transport protocol value.

[20] The one or more non-transitory computer-readable media of claim 15, further comprising dropping packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of data transfer comprises data corresponding to a particular real-time transport protocol value.

290. Sourcefire, in combination with the Emerging Threats rules, renders these claims obvious. Consider the following rule from the Emerging Threats rules:

```
#Blake Hartstein at Demarc
#
alert tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET
any (msg:"ET WEB_CLIENT Apple Quicktime RTSP Overflow
(1)"; flow:established,from_server; content:"|22|rtsp|3a|//";
nocase; isdataat:400,relative; content:!"|0a|"; distance:0;
within:400; content:!"|22|"; distance:0; within:400;
classtype:attempted-admin; reference:cve,2007-0015;
reference:bugtraq,21829;
reference:url,doc.emergingthreats.net/2003326;
reference:url,www.emergingthreats.net/cgi-
bin/cvsweb.cgi/sigs/WEB_CLIENT/WEB_Apple; sid:2003326;
rev:6;)
```

EX-1020 SID 2003326.

291. This rule is directed to detecting a buffer overflow attack "WEB_CLIENT Apple Quicktime RTSP Overflow". EX-1098. RTSP stands for "Real Time Streaming Protocol" and is "an application-level protocol for control over the delivery of data with real-time properties. RTSP provides an extensible framework to enable controlled, on-demand delivery of real-time data, such as audio and video." EX-1099 p. 1. Thus, RTSP is a "real time transport protocol". The buffer overflow exists in Apple Computer's Quicktime application, and specifically concerns overflows of the buffer for handling URLs for RTSP content. "By persuading a victim to open a specially-crafted QTL file or visit a

malicious Web page containing an overly long RTSP URL, a remote attacker could overflow a buffer and execute arbitrary code on the system." EX-1098.

292. I note that Real-Time Transfer Protocol (RTP) can refer to another particular transfer protocol for delivering audio and video over IP networks. However, a POSA would typically refer to that specific protocol by capitalizing the name and/or providing its common abbreviation, as done in the specification. EX-1001 9:4-5. However, the claims do not refer to a Real-Time Transfer Protocol value. Instead, they refer to "a particular real-time transport protocol value." A POSA would understand this term to refer to any particular real-time transport protocol value, which would include RTP, RTSP, and other protocols.

293. Rule 2003326 specifies a 5-tuple for packets returning from remote web servers to local clients ("tcp $EXTERNAL_NET $HTTP_PORTS -> $HOME_NET any"). If this matches, then in the operator, it looks for the content '|22|rtsp|3a|//'. Translating the hexadecimal, this is '"rtsp://', the beginning of a URL for transmission of data over RTSP.

294. As explained above in Section IX A, it would be obvious to a POSA to tune this rule to be a drop rule. If this is done, then testing for "rtsp://" would be "<u>comprising dropping packets associated with a particular type of data transfer responsive to a determination that the packets associated with the particular type of</u>

data transfer comprises data corresponding to a particular real-time transport protocol value", since the rule would block based on the evidence of a RTSP URL, thus preventing further transmission of that content.[10]

295. I note that the '343 patent discloses that a method to "detect if an IP packet contains a Real-time Transport Protocol" as an example of a use "for purposes other than network policy enforcement and exfiltration prevention." EX-1001 9:1-10. Thus, these dependent claims appear to recite limitations that are distinct from the packet transformation function configured to prevent an exfiltration operation, as discussed above in the context of the independent claims. As shown above, Emerging Threats discloses both rules for preventing exfiltration operations and for dropping packets associated with particular real-time transport protocol values, and both types of rules may be used with Sourcefire.

[10] The operator specified by rule 2003326 makes additional checks for the length and form of the URL, but to meet the "comprises" language in the claim, it is sufficient that the operator does check for the presence of an rtsp:// url as a means of checking for "data corresponding to a particular real-time protocol value".

X. Objective Indicia of Non-Obviousness

296. Based on my analysis of the '343 patent and its prosecution history, and my analysis of the other documents cited herein, and my extensive experience in the field prior to the earliest priority date of the '343 patent, I am not aware of any objective evidence that would suggest to me that any of the challenged claims are non-obvious. To the contrary, based on my experience and the evidence I address in this declaration, my opinion is that systems developed over a several year span before the priority date of the '343 patent clearly show that its claims were obvious to a POSA.

297. I understand that the Patent Owner argued that various evidence showed objective indicia of non-obviousness in the IPR of the '522 patent, but that the Board found that Patent Owner had not established the requisite nexus between the challenged claims in that IPR and any of the asserted secondary considerations, and the appeals court agreed. EX-1069 pp. 19-20. To the extent Patent Owner argues that secondary considerations are relevant to this proceeding, in my opinion they would not outweigh the strong evidence of obviousness of the '343 patent's claims as discussed above.

298. I have been advised that if Patent Owner presents evidence in this proceeding that it contends shows objective indicia of non-obviousness, I may be

asked to review such evidence and to offer my opinions. I reserve the right to assess and address any such evidence or arguments.

XI. Conclusion and Declaration

299. The findings and opinions set forth in this declaration are based on my work and examinations to date. I may continue my examinations. I may also receive additional documentation and other factual evidence over the course of this litigation that will allow me to supplement and/or refine my opinions. I reserve the right to add to, alter, or delete my opinions and my declaration upon discovery of any additional information. I reserve the right to make such changes as may be deemed necessary.

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

301. I declare that all statements made herein on my own knowledge are true and that all statements made on information and belief are believed to be true, and further, 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.

Executed on: 1/6/23

By: _____
Dr. Stuart Staniford