

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS, INC.
Petitioner,

v.

CENTRIPETAL NETWORKS, INC.
Patent Owner.

U.S. Patent No. 9,686,193 to Sean Moore
Issue Date: June 20, 2017
Title: Filtering Network Data Transfers

Inter Partes Review No.: Unassigned

**DECLARATION OF DR. STUART STANIFORD IN SUPPORT OF
PETITION FOR *INTER PARTES* REVIEW OF U.S. PATENT NO. 9,686,193**

Mail Stop “PATENT BOARD”

Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

CS-1003
Cisco Systems, Inc. v. Centripetal Networks, Inc.

**PAN EX1062
PAN v. Centripetal
IPR2023-01353
Page 00001**

TABLE OF CONTENTS

	Page
I. Introduction.....	1
II. List of Documents I Considered in Forming My Opinions.....	3
III. My Background and Qualifications.....	10
IV. Person of Ordinary Skill in the Art (POSA).....	14
V. Relevant Legal Standards	16
VI. The '193 Patent.....	19
A. Description	19
B. Priority Date of Claims	31
C. Prosecution History of the '193 Patent	31
VII. State of the Art.....	34
A. Packet-Switched Networking and Communication Protocols	35
B. Network Security Systems - Firewalls.....	37
C. Network Security Systems –Intrusion Detection Systems and Intrusion Prevention Systems.....	41
D. Examples of Instant Messaging and Multimedia Rules.....	45
E. Inspecting and Applying Packet Filtering Rules to Encrypted and Unencrypted SSL/TLS Record Protocol Data	53
VIII. Claim Construction.....	64
IX. Overview.....	65
X. Challenge: Claims 1-20 are Obvious under 35 U.S.C. § 103 over Sourcefire in View of the Knowledge and Skill of a POSA	67
A. Sourcefire Teaches Every Element of Claims 1-20 in View of the Knowledge and Skill of a POSA.....	67

1.	Sourcefire	67
B.	Detailed Invalidity Analysis.....	81
XI.	Conclusion and Declaration.....	121

I. Introduction

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

2. I have been retained by Cisco Systems, Inc. (“Cisco” 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 \$400 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. 9,686,193 (“the ’193 Patent”) (CS-1001). The ’193 Patent is titled “Filtering Network Data Transfers,” lists Sean Moore as the inventor, and is currently assigned to Centripetal Networks, Inc. (“Centripetal” or “Patent Owner”). The ’193 Patent issued June 20, 2017, from U.S. Patent Application No. 14/625,486 which was filed February 18, 2015, and is a continuation of and claims priority to U.S. Patent Application No. 13/795,822 filed on March 12, 2013, now U.S. Pat. No. 9,124,552. CS-1001; CS-1002.

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 ’193 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 '193 Patent is directed to systems and methods for filtering data packets leaving a network to ensure that the packets are not improperly transferring sensitive data or credentials from the network, a category of cyber-attacks known as exfiltrations. CS-1001, 1:14-36. I am familiar with the technology described in the '193 Patent as of the earliest possible priority date of March 12, 2013.

6. In preparing this Declaration, I have reviewed the '193 Patent (CS-1001), the file history of the '193 Patent (CS-1002), and each of the documents cited below, and 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 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 '193 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 '193 Patent. As described in detail below, I offer the following opinion in this Declaration:

- A POSA would have found the invention of Claims 1-20 of the '193 Patent to be obvious over the Sourcefire 3D System User Guide Version 4.10 (CS-1004) ("Sourcefire") in view of the knowledge, skill, and creativity of a POSA.

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 and other publications. I consider each of the references below as a reliable authority for the statements on which I rely.

Exhibit No.	Title
CS-1001	U.S. Patent No. 9,686,193 ("the '193 Patent")
CS-1002	Select portions of prosecution history of the '193 Patent ("File History")
CS-1003	Intentionally left blank.
CS-1004	Sourcefire 3D System User Guide Version 4.10 ("Sourcefire")
CS-1005	Declaration of John Leone
CS-1006	U.S. Patent No. 7,032,031 ("Jungck")
CS-1007	U.S. Patent No. 5,835,726 ("Shwed")
CS-1008	U.S. Patent No. 8,370,936 ("Zuk")
CS-1009	U.S. Patent No. 7,900,042 ("Hall")

Exhibit No.	Title
CS-1010	SSL Inspection – Sourcefire Cybersecurity Webpage, archived at Internet Archive Wayback Machine on December 20, 2010 at https://web.archive.org/web/20101220151851/http://www.sourcefire.com/security-technologies/cyber-security-products/3d-system/ssl-encryption-decryption (“SSL Appliances Webpage”)
CS-1011	Sourcefire SSL Appliances data sheet, archived at Internet Archive Wayback Machine on March 16, 2012 at https://web.archive.org/web/20120316165725/http://www.ndm.net:80/sourcefirestore/downloads/Sourcefire%20SSL%20Appliance%20Fact%20Sheet%20%28English%29.pdf (“SSL Appliances Data Sheet”)
CS-1012	Intentionally left blank.
CS-1013	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
CS-1014	Bard, G.V., “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/20060503043634/http://eprint.iacr.org:80/2006/136.pdf
CS-1015	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
CS-1016	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

Exhibit No.	Title
CS-1017	NDM Company (“Sourcefire Authorized Distributor”) Product Data Sheets Webpage, archived at Internet Archive Wayback Machine on October 28, 2011 at https://web.archive.org/web/20111028104123/http://www.ndm.net:80/sourcefirestore/Product-Data-Sheets/product-data-sheets (“NDM Webpage”)
CS-1018	A. Shostack, “An Overview of SSL (version 2),” May 1995, available from https://adam.shostack.org/ssl.html (accessed July 11, 2018), archived at Internet Archive Wayback Machine on May 25, 1997 at https://web.archive.org/web/19970525224610/http://www.homeport.org/~adam/ssl.html .
CS-1019	M. Bellovin and William R. Cheswick, “Network Firewalls,” <i>IEEE Communications Magazine</i> , September 1994
CS-1020	Barry M. Leiner <i>et al.</i> , “A Brief History of the Internet”. <i>Internet Society</i> , 1997, available at: https://www.internetsociety.org/internet/history-internet/brief-history-internet/ (accessed June 18, 2018).
CS-1021	“Internet Protocol,” <i>IETF RFC 791</i> , J. Postel, ed., September 1981, archived at Internet Archive Wayback Machine on February 4, 2007 at https://web.archive.org/web/20070204151303/https://tools.ietf.org/html/rfc791 .
CS-1022	“Transmission Control Protocol,” <i>IETF RFC 793</i> . 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 .
CS-1023	E. H. Spafford, “The Internet Worm Program: An Analysis,” <i>Purdue Technical Report CSD-TR-823</i> , 1988.

Exhibit No.	Title
CS-1024	Eichin, M.W. and Rochlis, J.A., “With Microscope and Tweezers: The Worm from MIT’s Perspective,” <i>Communications of the ACM</i> , Volume 32 Number 6, June 1989.
CS-1025	T. Eisenberg <i>et al.</i> , “The Cornell Commission: On Morris and the Worm,” <i>Communications of the ACM</i> , Volume 32 Number 6, June 1989.
CS-1026	J. C. Mogul, “Simple and Flexible Datagram Access Controls for Unix-based Gateways,” <i>Digital Western Research Laboratory Technical Report No. 89/4</i> , 1989.
CS-1027	Heberlein, L. T.L, <i>et al.</i> , “A Network Security Monitor,” <i>Proceedings, 1990 IEEE Computer Society Symposium on Security and Privacy</i> , 1990.
CS-1028	“NetRanger Network Security Management System User Guide”. v1.3.1 <i>Wheelgroup Corporation</i> , 1997.
CS-1029	Roesch, M., “Snort: Lightweight Intrusion Detection for Networks,” <i>Lisa</i> (Vol. 99, No. 1, pp. 229-238), 1999.
CS-1030	Brandon De Hoyos, “IRC, ICQ, AIM and More: A History of Instant Messaging,” <i>Lifewire</i> , Oct 2016, available from https://www.lifewire.com/im-a-brief-history-1949611 (accessed July 10, 2018).
CS-1031	M. Chatel, “Classical vs. Transparent IP Proxies,” <i>IETF RFC 1919</i> , March 1996 archived at Internet Archive Wayback Machine on June 30, 2007 at https://web.archive.org/web/20070630145311/https://tools.ietf.org/html/rfc1919 .
CS-1032	K. Ingham & S. Forrest, “A History and Survey of Network Firewalls,” <i>University of New Mexico Computer Science Department Technical Report</i> archived at Internet Archive Wayback Machine on September 2, 2006 at https://web.archive.org/web/20060902171316/https://www.cs.unm.edu/~treport/tr/02-12/firewall.pdf .

Exhibit No.	Title
CS-1033	Sourcefire Support Website Webpage, archived at Internet Archive Wayback Machine on April 30, 2011 at https://web.archive.org/web/20110430194135/https://support.sourcefire.com/ (“Sourcefire Support Webpage”)
CS-1034	Business Wire, “Sourcefire Expands IPS Solutions Portfolio, Adding FirePOWER to the Industry’s Leading Protection,” April 18, 2011, available at https://www.businesswire.com/news/home/20110418005189/en/Sourcefire-Expands-IPS-Solutions-Portfolio-Adding-FirePOWER (accessed June 20, 2018)
CS-1035	“File Transfer Protocol,” <i>IETF RFC 765</i> , J. Postel, ed., June 1980, archived at Internet Archive Wayback Machine on February 6, 2007 at https://web.archive.org/web/20070206005843/https://tools.ietf.org/html/rfc765 .
CS-1036	T. B. Lee, “The Original HTTP as defined in 1991,” <i>World Wide Web Consortium</i> , 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 .
CS-1037	I. Ristic, “SSL/TLS and PKI History,” April 2018, available from https://www.feistyduck.com/ssl-tls-and-pki-history/ , (accessed June 27 th , 2018).
CS-1038	E. Rescorla, “HTTP over TLS”, <i>IETF RFC 2818</i> , 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 .
CS-1039	J. Moser, “The First Few Milliseconds of an HTTPS Connection”, <i>Moserware</i> , June 10, 2009, archived at Internet Archive Wayback Machine on June 13, 2009 at https://web.archive.org/web/20090613041022/http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html .

Exhibit No.	Title
CS-1040	M. Allman, et al., “FTP Security Considerations,” IETF RFC 2577, May 1999, archived at Internet Archive Wayback Machine on March 18, 2007 at https://web.archive.org/web/20070318022322/https://tools.ietf.org/html/rfc2577 .
CS-1041	H. Schulzrinne, et al., “RTP: A Transport Protocol for Real-Time Applications,” IETF RFC 3550, July 2003, archived at Internet Archive Wayback Machine on February 6, 2007 at https://web.archive.org/web/20070206011723/https://tools.ietf.org/html/rfc3550 .
CS-1042	Sourcefire, Inc., “Chat.rules” file, April 2005, downloaded from “Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release) . . . VRT Certified Rules for Snort v2.3,” archived at Internet Archive Wayback Machine on January 12, 2006 at https://web.archive.org/web/20060112063540/http://www.snort.org:80/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.3.tar.gz .
CS-1043	Sourcefire Research Team, “2450.txt” file, April 2005, downloaded from “Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release) . . . VRT Certified Rules for Snort v2.3,” archived at Internet Archive Wayback Machine on January 12, 2006 at https://web.archive.org/web/20060112063540/http://www.snort.org:80/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.3.tar.gz .
CS-1044	P. Saint Andre, Ed., “Extensible Messaging and Presence Protocol (XMPP): Core,” <i>Network Working Group RFC 3920</i> , October 2004, archived at Internet Archive Wayback Machine on October 10, 2006 at https://web.archive.org/web/20061010092431/https://xmpp.org/rfc/rfc3920.html .

Exhibit No.	Title
CS-1045	P. Saint Andre, Ed., “Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence,” <i>Network Working Group RFC 3921</i> , October 2004, archived at Internet Archive Wayback Machine on October 10, 2006 at https://web.archive.org/web/20061010092547/http://xmpp.org:80/rfc/rfc3921.html .
CS-1046	R. Fielding, et. al., “Hypertext Transfer Protocol -- HTTP/1.1,” <i>Network Working Group RFC 2616</i> , June 1999, archived at Internet Archive Wayback Machine on August 15, 2000 at https://web.archive.org/web/20000815082940/https://www.ietf.org/rfc/rfc2616.txt .
CS-1047	“XMPPLoat – A Tool to Attack XMPP Connections,” August 27, 2012, archived at Internet Archive Wayback Machine on August 29, 2012 at https://web.archive.org/web/20120829042816/https://www.darknet.org.uk/2012/08/xmpploit-a-tool-to-attack-xmpp-connections/ .
CS-1048	Sourcefire, Inc., “Multimedia.rules” file, April 2005, downloaded from “Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release) . . . VRT Certified Rules for Snort v2.3,” archived at Internet Archive Wayback Machine on January 12, 2006 at https://web.archive.org/web/20060112063540/http://www.snort.org:80/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.3.tar.gz .
CS-1049	Sourcefire Research Team, “1437.txt” file, April 2005, downloaded from “Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release) . . . VRT Certified Rules for Snort v2.3,” archived at Internet Archive Wayback Machine on January 12, 2006 at https://web.archive.org/web/20060112063540/http://www.snort.org:80/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.3.tar.gz .

Exhibit No.	Title
CS-1050	H. Schulzrinne, et al., “RTP Profile for Audio and Video Conferences with Minimal Control,” <i>IETF RFC 3551</i> , July 2003, archived at Internet Archive Wayback Machine on February 6, 2007 at https://web.archive.org/web/20070206011733/https://tools.ietf.org/html/rfc3551 .
CS-1051	J. Rosenberg, et al., “SIP: Session Initiation Protocol,” <i>IETF RFC 3261</i> , June 2002, archived at Internet Archive Wayback Machine on April 28, 2006 at https://web.archive.org/web/20060428031957/https://tools.ietf.org/html/rfc3261 .
CS-1052	M. Handley, et al., “SDP: Session Description Protocol,” <i>IETF RFC 4566</i> , July 2006, archived at Internet Archive Wayback Machine on January 28, 2007 at https://web.archive.org/web/20070128163222/http://tools.ietf.org:80/html/rfc4566 .

III. My Background and Qualifications

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

10. 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 as Exhibit A, and it provides a comprehensive description of my

background, experience, and academic and employment history, including a list of network security publications for at least the last 26 years, and a list of cases in which I have testified as an expert witness at trial or by deposition.

11. 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 Masters of Science in Computer Science from UC Davis, having transferred into the network security group there. My Masters’ thesis focused on a technique for tracing computer attackers back to their true location on the Internet.

12. 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.

13. 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 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.

14. 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.

15. 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.

16. 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.

17. From 2013 to 2016 I was an Adjunct Professor at Cornell University. I taught a yearly graduate course on applied network security.

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

19. 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.

20. I have co-authored a number of publications in the network security area, which have attracted over 7000 citations in the academic literature. I am a named inventor on three patents, U.S. Patent Nos 8,850,571, 8,997,219, and 9,106,694.

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

21. I am familiar with the knowledge and capabilities of one of ordinary skill in the art. Through my work as a researcher, professor, network security system designer, architect and software engineer, and founder, chief scientist, and/or principal scientist of several network security companies, my direct participation (including as chairman) of standards-setting working groups, and my direct collaborations with industry designers, architects, and engineers during that period, I have become personally familiar with the level of skill of individuals in the relevant field and the general state of the art. Unless otherwise stated, my testimony below refers to the knowledge of one of ordinary skill in the art as of March 2013. See also Section I ¶6; Section VI.B below.

22. I have been informed and understand that a Person of Ordinary Skill in the Art (“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.

23. With respect to the ’193 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.

24. 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.

V. Relevant Legal Standards

25. 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 '193 Patent are anticipated or 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.

26. I have been informed and understand that, to anticipate a claim under 35 U.S.C. § 102, a reference must teach every element of the claim either expressly or inherently to a person having ordinary skill in the relevant art.

27. Further, 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.

28. I understand that objective indicia 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.

29. 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.

30. 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.

31. 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.

32. 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 '193 Patent

A. Description

33. The '193 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 a casual 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. CS-1001 1:24-47.

34. The '193 Patent describes a two-stage filtering process for addressing such attacks implemented at a Packet Security Gateway (PSG) examining packets that are exiting a trusted network towards an external network. CS-1001 8:39-52.

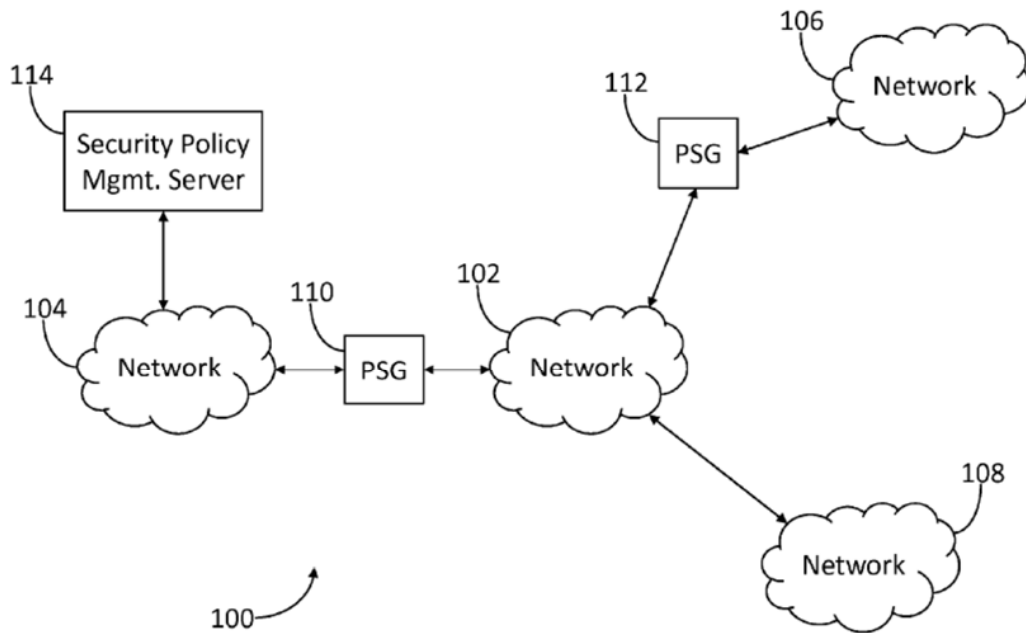


FIG. 1

35. FIG. 1 of the '193 Patent (above) illustrates a network environment 100 of various networks protected by PSGs 110, 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. CS-1001 3:27-41. Networks 104 and 106 are protected by PSGs 110, 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. CS-1001 3:58-4:23; 5:25-45.

36. 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. CS-1001 4:23-37.

37. 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. CS-1001 8:39-51.

38. FIG. 4 of the '193 Patent (below) illustrates the two-stage filtering process performed at a PSG:

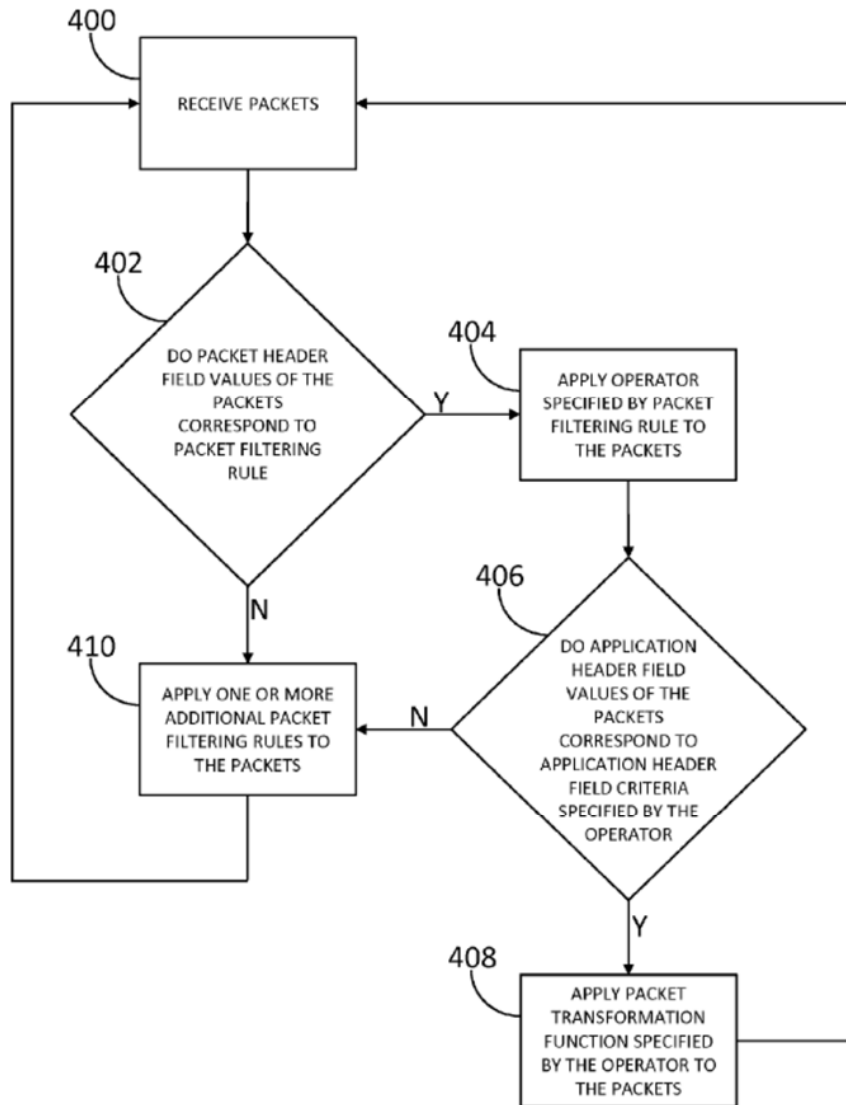


FIG. 4

39. 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. CS-1001 9:19-30.

40. 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. CS-1001 9:30-35.

FIG. 3 of the '193 Patent (below) identifies example operators that may be specified by a rule for packet header field values.

41. 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 rule. CS-1001 9:38-42.

42. 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 rule is applied to the received packets. Notwithstanding the name “packet transformation function”, the '193 Patent describes that this function can include encapsulate the packets into a tunnel, alter one or more header field values,

as well as allow or block transmission of the packets. CS-1001 7:41-45; 9:47-61; Claims 1, 2.

43. In one embodiment, the '193 Patent describes that an operator specified by a rule 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). CS-1001 2:36-43; FIG. 3 (below) (rule 6).

44. FIG. 3 of the '193 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

45. 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. CS-1001 5:46-60.

46. 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 e-mail into an email client). CS-1001 5:61-6:21.

47. 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. CS-1001 6:22-30. 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.” CS-1001 6:57-67, 7:38-45, 8:9-38. In the specific embodiment disclosed for Rule 5, the ’193 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. CS-1001 8:11-16.

48. 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) should have an HTTP-EXFIL operator applied to them in a second stage. 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 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 ’193 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.” CS-1001 7:1-20.

49. 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. CS-1001 6:31-36.

50. 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, alter one or more header field values in the packets, etc. CS-1001 7:38-45, 9:47-61.

51. By enforcing sets of rules similar to rules 1-7, the '193 Patent states 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). CS-1001 7:32-37.

52. The claims of the '193 Patent are directed to a method, system and computer program for implementing particular combinations of packet-filtering rules at the PSG. Independent Claim 1 is directed to the method; independent Claim 18 is a corresponding system claim performing the steps recited in Claim 1; and independent Claim 19 is a corresponding computer program programmed to

perform the steps recited in Claim 1. Independent Claim 8 is a method Claim similar to Claim 1 except that the packet filtering rules are applied to packets that are destined to the same network, while in the other independent claims the packet filtering rules are applied to packets destined to two different networks. Therefore, Claim 1 is representative of the independent claims:

1. A method comprising:

receiving, by a computing system and from a computing device located in a first network, a plurality of packets, wherein the plurality of packets comprises a first portion of packets and a second portion of packets;

responsive to a determination by the computing system that the first portion of packets comprises data corresponding to criteria specified by one or more packet-filtering rules configured to prevent a particular type of data transfer from the first network to a second network, wherein the data indicates that the first portion of packets is destined for the second network;

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; and

dropping, by the computing system, each packet in first portion of packets; and

responsive to a determination by the computing system that the second portion of packets comprises data that does not correspond to

the criteria wherein the data indicates that the second portion of packets is destined for a third network;

applying, by the computing system and to each packet in the second portion of packets, and without applying the one or more packet-filtering rules configured to prevent the particular type of data transfer from the first network to the second network, a second operator configured to forward packets not associated with the particular type of data transfer toward the third network; and

forwarding, by the computing system, each packet in the second portion of packets toward the third network.

53. Claim 1, while verbose, is directed to providing two rules, one for “dropping” (*i.e.*, blocking) packets and one for “forwarding” (*i.e.*, allowing) packets. The first rule (recited in the first “responsive” element of Claim 1) is crafted to “drop” packets associated with a defined type of data transfer (*e.g.*, HTTP, HTTPS, FTP, or RTP protocols, methods/commands/values within such protocol, etc.) that are going from a first network (*e.g.*, internal network) to a second network (*e.g.*, external network). The second rule (recited in the second “responsive” element of Claim 1) is crafted to “forward” packets that are going from the first network to a different network (the “third network”) where those packets are *not* associated with the type of data transfer that was blocked by the first rule.

54. Claim 8 differs from Claim 1 in two respects: (i) the first rule is crafted to “drop” packets associated with a defined type of messaging protocol (XMPP) rather than a defined type of data transfer, and (ii) both rules operate on packets destined for the same network (the “second network”).

55. The dependent claims essentially recite that the “drop” rule (*i.e.*, the first rule) can be directed to different protocols, including HTTP (claim 4), HTTPS (claim 5), FTP (claim 6) or RTP (claim 7).

B. Priority Date of Claims

56. As I discuss in Section VI.C, the ’193 Patent (CS-1001) issued from U.S. Application No. 14/625,486 filed February 18, 2015 (CS-1002) and is a continuation of and claims priority to U.S. Application No. 13/795,822 filed on March 12, 2013, now U.S. Pat. No. 9,124,552. CS-1002 at pp. 1-41; CS-1001 at p. 1.

57. For the purposes of this *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. Prosecution History of the ’193 Patent

58. Prior to examination by the USPTO, the originally filed claims were cancelled and replaced by new, additional claims. The independent claims of these replacement claims were directed to applying two rules, one for “dropping” (*i.e.*,

blocking) packets and one for “forwarding” (*i.e.*, allowing) packets, to packets being transferred from a first network to the same second network. In the first Office Action, the examiner rejected the independent claims over prior art, but indicated the allowability of (1) a dependent claim that recited that the second rule operated on packets that were destined to a different “third network”, (2) a dependent claim that recited a “dropping” rule directed to the XMPP type of data transfer, and (3) dependent claims that recited allowing or dropping a packets based on “a particular transport layer security (TLS) version value.” CS-1002 pp. 17-23, 46-56, 157-181.

59. Applicant then amended the independent claims to include the limitations recited in the allowed dependent claims that recited (1) the “third network” and (2) dropping XMPP packets. In response to these amendments, a notice of allowance issued. CS-1002 pp. 271-283, 292-294.

60. With respect to independent claims 1, 18, and 19, although the USPTO examiner found patentability in the particular combination of (1) a rule specifying “dropping” packets transiting from a first network to a second network and that are associated with a particular type of data transfer (*e.g.*, HTTP, FTP, RTP protocols), and (2) another rule specifying “allowing” packets transiting from the first network to a third network and that are *not* associated with the particular type of data transfer (CS-1002 pp. 46-56, 160, 271-283, 292-294), the ’193 Patent

does not make any distinction that the packet-filtering use of either rules' criteria, or particular combinations of rules specifying such criteria, is more technologically challenging than, for example, use of other application layer rule criteria such as encryption protocol (*e.g.*, SSL, TLS versions), or other combinations of different rules. CS-1001 6:58-9:10; See also Section VI.A.

61. Rather, the '193 Patent treats all application layer rule criteria the same, and all combinations of rules the same, and identifies *the exact same system architecture and method* for allowing or blocking received packets using rule criteria, and combinations of rule criteria, specifying each of the different application layer data types, any possible destination, and any possible origin. See also Section VI.A.

62. Likewise, with respect to Claim 8, although the USPTO examiner found patentability in the particular combination of (1) a rule specifying “dropping” packets transiting from a first network to a second network and that are associated with a particular type of messaging protocol (XMPP), and (2) another rule specifying “allowing” packets transiting from the first network to the same second network and that are *not* associated with the XMPP (CS-1002 pp. 46-56, 160, 271-283, 292-294), the '193 Patent does not make any distinction that particular use of either rules' criteria for packets being transferred to the same network is more technologically challenging than, for example, use of other

application layer rule criteria such as encryption protocol (*e.g.*, SSL, TLS versions), or other combinations of different rules. CS-1001 6:58-9:10; See also Section VI.A.

63. Indeed, the '193 Patent describes that the exact same system architecture and method is used to allow or block packets with application layer data specifying certain (i) data *transfer* protocols (*e.g.*, HTTP, FTP, RTP), (ii) data *messaging* protocols (*e.g.*, XMPP), and (iii) data *encryption* protocols (*e.g.*, TLS 1.0). CS-1001 6:58-9:10; See also Section VI.A. Moreover, the '193 Patent describes that the exact same system architecture and method is used to allow or block packets with packet header data specifying any possible destination and any possible origin. See also Section VI.A.

VII. State of the Art

64. The following 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 and including 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 of the invention of the '193 Patent, provide additional motivation to modify or combine the teachings of the Sourcefire reference (CS-1004) with the knowledge, skill, and creativity of a POSA, provide

support for why a POSA would have had a reasonable expectation of success in doing so, rebut any assertion by the Patent Owner of unpredictability in the art, and rebut any assertion by the Patent Owner of unexpected results.

65. 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: packet-switched networking, firewalls, security policies, communication protocols and layers, and the use of customized rules to address cyber-attacks. See also Section IV.

A. Packet-Switched Networking and Communication Protocols

66. 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. CS-1020. The early ARPANET used a protocol, NCP, that is no longer in use and proved not to be scalable.

67. 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 (CS-1022), and Transmission Control Protocol (TCP) to

provide reliable delivery of streams of data across an IP internetwork (CS-1021). These two protocols were described in the IETF Request For Comment (RFC) documents RFC 791 and 793 respectively in 1981. CS-1022; CS-1021. The ARPANET switched 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. CS-1020.

68. Logically, the Internet is a web of intermediate network nodes and communication paths interconnecting those network nodes that provides multiple data transmission routes from any given point to any other given point on the network (*i.e.*, between any two computers.) CS-1006 3:59-4:11. The disparate networks are connected where they are joined by special devices – IP routers (often known in early literature as gateways) – 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. CS-1006 3:32-46, 38:29-39:8.

69. 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. CS-1022; CS-1021; CS-1020. 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.

CS-1020. Other important protocols included FTP for making files available, first defined in 1980 (CS-1035), the Hypertext Transfer Protocol (HTTP) for facilitating the exchange of hypertext markup language (HTML) and World Wide Web (WWW) content, first defined in 1991 (CS-1036). Similarly, Telnet and then SSH were developed and were well-known for remote computer terminal access, as were a variety of instant messaging protocols that became available for text-based communication (*e.g.*, AOL Instant Messenger (AIM) and ICQ, Windows Live Messenger, Yahoo! Messenger, Jabber (using XMPP protocol (CS-1044; CS-1045), etc.). CS-1030.

70. Protocols were also developed for providing voice/video telephone and conferencing service (Voice-over-IP or VOIP), originally in the late 1990s. Examples include Session Initiation Protocol (SIP) for managing the setting up and tearing down of calls (CS-1051), Session Description Protocol (SDP) for specifying which media format was to be used on a call (CS-1052), and the Real-Time Protocol (RTP) for actually carrying the audio/video streams that constituted a call (CS-1041; CS-1050).

B. Network Security Systems - Firewalls

71. 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). CS-1020.

72. This era of complete openness spawned a variety of abuses by computer hackers. A particularly spectacular and famous incident spawned 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 or IDS) were under development. CS-1023, CS-1024, CS-1025.

73. 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.

74. It was well-known to include firewall functionality in some routers to intercept and review packets for security purposes. Indeed, routers provided 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.” CS-1026 p. 3. This example illustrates that the basic functionality of blocking or admitting packets based on a 5-tuple was available prior to 1989.

75. From early on, it was apparent that some computer applications required special handling. Thus, for example, the first commercial firewall, the DEC SEAL released in 1991, used a packet filter to control what packets could enter or leave the network but also required most application traffic to transit an explicit application proxy server. CS-1032 p. 9. Here, an internal network client desiring to access exterior network resources would make a TCP connection to the exterior application server (and the packet filter would be configured to allow this, but not direct outbound connections). The client would then authenticate to the application proxy, permissions would be checked and, if authorization was granted, the client connection was continued and the proxy would make a second TCP

connection to the external server. The proxy would then relay requests/responses in the application back and forth, while logging and possibly filtering the traffic. CS-1032 pp. 8-11.

76. 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). These issues 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. CS-1032; CS-1008 4:12-30.

77. By the mid-1990s, multiple systems had been developed which incorporated application functionality "transparently" into stateful inspection firewalls. CS-1031 (published in 1996). The way this functionality 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. CS-1008 4:31-41.

78. Based on understanding the application protocol, the firewall determined that certain TCP connections needed to be routed through a proxy. The firewall 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 not only the 5-tuple (*e.g.*, ports and IP addresses) but also application-layer data in the packet data bodies.

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

79. 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. CS-1027. This evolved into the 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.

80. 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, the 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[]+"

CS-1028 p. 189.

81. 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. CS-1028 p. 189.

82. 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.” CS-1046 pp. 44, 46-50, 130-134.

83. The first widely adopted open source network intrusion detection system (NIDS or just IDS), Snort, was first released in 1998, and was documented in a conference paper in 1999. CS-1029. 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!";)
```

84. This rule (*see* CS-1029 p 232, 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) matched 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.

85. 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. CS-1028 p. 4-26.

86. The need to block attacks, as well as the need to resolve various complex ambiguities in the meaning of packet traffic that computer intruders used to get around the first NIDS, lead to the industry quickly developed systems that were "inline" with 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).

87. 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, described 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. CS-1008 col. 18, Fig. 11.

D. Examples of Instant Messaging and Multimedia Rules

88. An example of other content keywords in a rule includes the following Snort rule that alerted the user based on TCP protocol packets from any source IP address on an external network and port 5050 (characteristic of Yahoo! Messenger traffic) to any destination IP address in the internal network monitored by Snort and any destination port:

```
alert tcp $EXTERNAL_NET 5050 -> $HOME_NET any (msg:"CHAT Yahoo IM successful logon";  
flow:from_server,established; content:"YMSG"; depth:4; nocase; content:"|00 01|";  
depth:2; offset:10; classtype:policy-violation; sid:2450; rev:3;)
```

CS-1042, p. 2. The particular content keyword (YMSG) matches content indicative of an internal host's successful attempt to login to the popular Yahoo! Messenger service. *Id.* The rule (CS-1042, p. 2) identified that such action would violate corporate policy and the corresponding signature document for the rule (CS-1043) further detailed that:

Instant Messenger programs may not be appropriate in certain network environments . . . A user must successfully logon to an Yahoo Instant Messenger server before participating in any exchanges, such sending or receiving messages, files, or webcams, or chatting by voice. Many of these activities are not appropriate in a corporate environment. Also, the exchanges are transacted via Yahoo IM servers so there is no assurance of privacy . . . Attack Scenarios: Once logged in, a Yahoo IM user may unwittingly accept a malicious file that may contain a worm, virus, Trojan, or backdoor to name a few. Ease of Attack: Easy . . . Corrective Action: Disallow the use of IM clients on the protected network and enforce or implement an organization wide policy on the use of IM clients.

CS-1043 pp. 1-2. Vulnerabilities with various IM protocols, including Jabber (using XMPP), were known to a POSA, and thus a POSA understood that enforcing security policies which disallowed the use of all known IM protocols on a protected network was a straightforward way to protect an internal network from such vulnerabilities. CS-1043 pp. 1-2; CS-1044 pp. 41-44; CS-1045 p. 58; CS-1047.

89. As another example, the below Snort rule alerted the user based on TCP protocol packets from any source IP address on an external network and port

80 to any destination IP address in the internal network monitored by Snort and any destination port:

```
alert tcp $EXTERNAL_NET 80 -> $HOME_NET any (msg:"MULTIMEDIA Windows Media
download"; flow:from_server,established; content:"Content-Type|3A|"; nocase;
pcre:"/^Content-Type\x3a\s*(?=[av])(video/x\-\ms\-(w[vm]x|asf)|a(udio/x\-\ms\-(m[av]
)|ax)|pplication/x\-\ms\-\wm[zd]))/smi"; classtype:policy-violation; sid:1437;
rev:6;)
```

CS-1048 p. 1. The particular content keyword matched content indicative of an internal host's use of the popular Windows Media application. CS-1048 p. 1. The rule (CS-1048 p. 1) identified that such action would violate corporate policy and the corresponding signature document for the rule (CS-1048) further detailed that:

This event is generated when network traffic indicating the use of a multimedia application is detected . . . This may be a violation of corporate policy since these applications can be used to bypass security measures designed to restrict the flow of corporate information to destinations external to the corporation . . . Multimedia client applications can be used to view movies and listen to music files. Some also include file sharing facilities. Use of these programs may constitute a violation of company policy. Clients may also contain vulnerabilities that can give an attacker an attack vector for delivering Trojan horse programs and viruses . . . Attack Scenarios: A user can download files from a source external to the protected network that may contain malicious code hidden in the file giving an

attacker the opportunity to gain access to a host inside the protected network. Ease of Attack: Simple.

CS-1049, pp. 1-2. Vulnerabilities with various multimedia transfer protocols including RTP were known to a POSA, and thus a POSA understood that enforcing security policies which disallowed the use of such protocols on a protected network was a straightforward way to protect an internal network from such vulnerabilities. CS-1049, pp. 1-2; CS-1041 p. 51; CS-1050 p. 27.

90. CS-1042, CS-1043, CS-1048 and CS-1049 are printouts of archived files included in the “VRT Certified Rules for Snort v2.3” archived file that I downloaded from archived web pages on the publicly available Snort community website that are currently available from the Internet Archive’s Wayback Machine.

91. Specifically, I downloaded the “VRT Certified Rules for Snort v2.3” file as a compressed file (snortrules-pr-2.3.tar.gz) from the “Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release)” section of the “Download Rules” web page (screenshots reproduced below):

Internet Archive
Wayback Machine
366 captures
12 Mar 2005 - 20 Jun 2017

Secure https://web.archive.org/web/20050810025529/http://www.snort.org:80/pub-bin/downloads.cgi

JUL AUG SEP 10 2005 2006

GO

SET SOURCE? | ABOUT SNORT | ABOUT SOURCEFIRE | FAQ

developed by SOURCEfire

NEWS
DOWNLOAD
DOCS
RULES
COMMUNITY
FORUMS
TRAINING

Search Rules

Account
email
password
LOGIN
NOT REGISTERED?
USER PREFERENCES

DOWNLOAD RULES

[Sourcefire VRT Certified Rules \(subscription release\)](#) | [Sourcefire VRT Certified Rules](#) | [Sourcefire VRT Certified Rules \(unregistered user release\)](#) | [Community Snort Rules](#)

Sourcefire VRT Certified Rules - The Official Snort Ruleset (subscription release)

These rules are available to subscribers only until Wednesday, July 27th 2005

The Sourcefire Vulnerability Research Team (VRT) has learned of serious vulnerabilities affecting Microsoft Windows, RealPlayer, MailEnable, the PHP XML-RPC module and FutureSoft TFTP server.

[view advisory](#) | [view changelog](#)

**Snort v2.2 users will need to update to the latest snort src snapshot located at <http://www.snort.org/snapshot-current-2.2.0.tar.gz> or set your snort.conf to "config flowbits_size: 64".

Snort 2.4 or higher is required to run CURRENT

VRT Certified Rules for Snort CURRENT (subscription release)	(snortrules-snapshot-CURRENT.tar.gz)	Download
MD5: a055e04a8b08840e0b07ae1ff7aab0d6	VER: CURRENT RELEASED: 2005-07-22	
VRT Certified Rules for Snort v2.4 (subscription release)	(snortrules-snapshot-2.4_s.tar.gz)	Download
MD5: 35d9a24868b0c280b6493aa03c011927	VER: 2.4 RELEASED: 2005-07-27	
VRT Certified Rules for Snort v2.3 (subscription release)	(snortrules-snapshot-2.3_s.tar.gz)	Download
MD5: c7243ba00332a5a9494a1540d9495f2	VER: 2.3 RELEASED: 2005-07-22	
VRT Certified Rules for Snort v2.2 (subscription release)	(snortrules-snapshot-2.2_s.tar.gz)	Download
MD5: a09b30a05977ac02d47863163f9680	VER: 2.2 RELEASED: 2005-07-22	
VRT Certified Rules for Snort v2.1 (subscription release)	(snortrules-snapshot-2.1_s.tar.gz)	Download
MD5: b766a09054ca5d70b19a021ca3dc08	VER: 2.1 RELEASED: 2005-07-22	
VRT Certified Rules for Snort v2.0 (subscription release)	(snortrules-snapshot-2.0_s.tar.gz)	Download
MD5: 573a2a00d0083c513607b2b8b6bd40a82	VER: 2.0 RELEASED: 2005-07-22	

Sourcefire VRT Certified Rules - The Official Snort Ruleset (unregistered user release)

The Sourcefire Vulnerability Research Team (VRT) has learned of serious vulnerabilities affecting Microsoft Windows, RealPlayer, MailEnable, the PHP XML-RPC module and FutureSoft TFTP server.

[view advisory](#) | [view changelog](#)

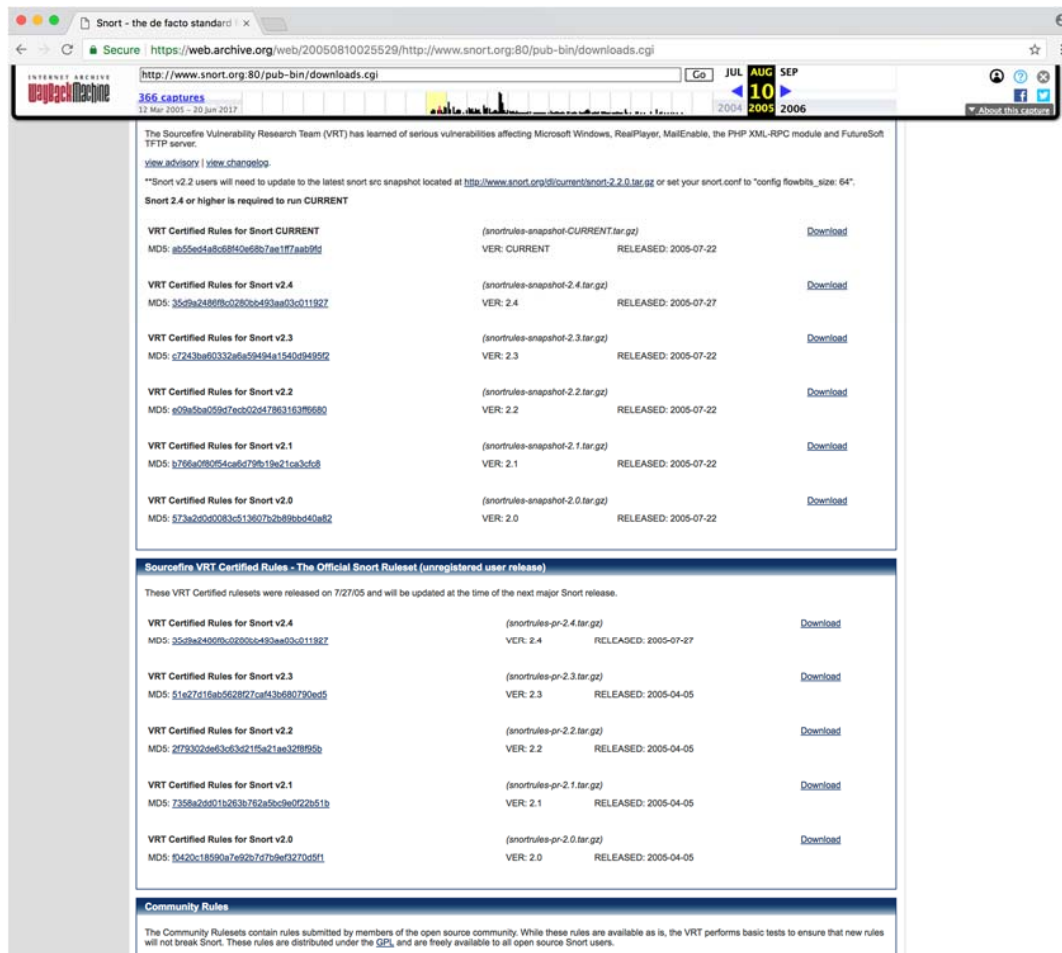
**Snort v2.2 users will need to update to the latest snort src snapshot located at <http://www.snort.org/snapshot-current-2.2.0.tar.gz> or set your snort.conf to "config flowbits_size: 64".

Snort 2.4 or higher is required to run CURRENT

VRT Certified Rules for Snort CURRENT	(snortrules-snapshot-CURRENT.tar.gz)	Download
MD5: a055e04a8b08840e0b07ae1ff7aab0d6	VER: CURRENT RELEASED: 2005-07-22	
VRT Certified Rules for Snort v2.4	(snortrules-snapshot-2.4.tar.gz)	Download
MD5: 35d9a24868b0c280b6493aa03c011927	VER: 2.4 RELEASED: 2005-07-27	

Sourcefire VRT Certified Rules are the official rules of snort.org. Each rule has been rigorously tested against the same standards the VRT uses for Sourcefire customers.

Trouble downloading a file?
Please contact us at snort-rt@sourcefire.com



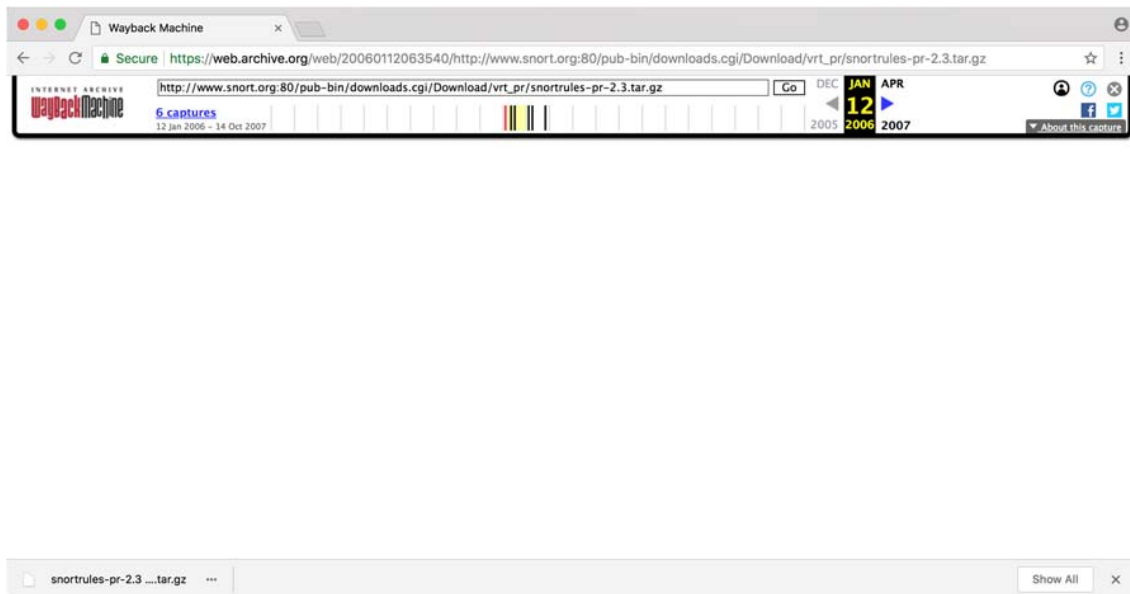
92. As illustrated in the screenshots above, I understand that the “Download Rules” web page (<http://www.snort.org:80/pub-bin/downloads.cgi>), is currently publicly accessible through the Internet Archive’s Wayback Machine at: <https://web.archive.org/web/20050810025529/http://www.snort.org:80/pub-bin/downloads.cgi>.

93. I have read and understand from the Frequently Asked Questions (“FAQ”) web page (<https://archive.org/about/faqs.php>) for the Internet Archive’s Wayback Machine that the list of numbers in the middle of an archived uniform

resource location (“URL”) are to be translated as yyyyymmddhhmmss. I therefore understand that the above-reproduced web page was archived by the Internet Archive on August 10, 2005.

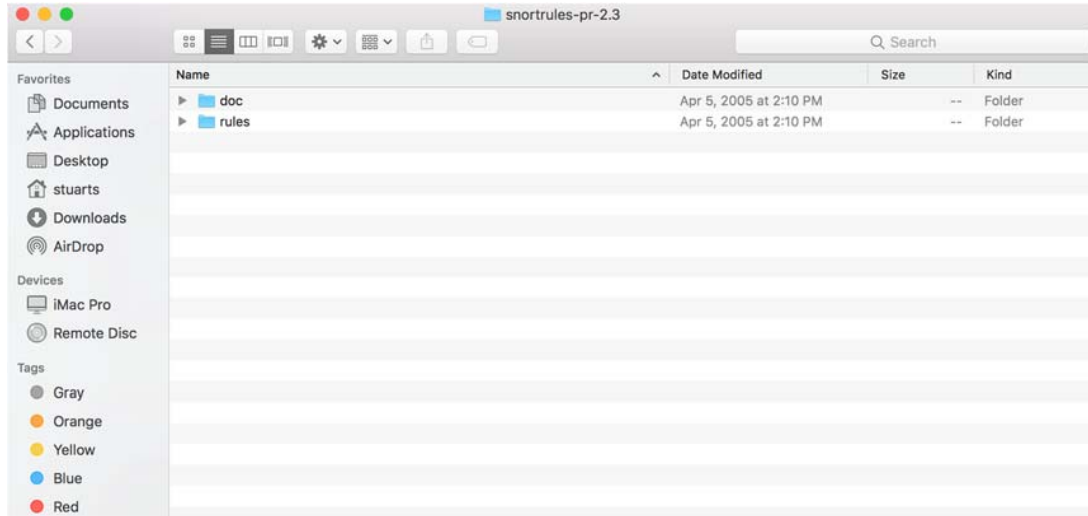
94. As illustrated in the screenshot below, I also understand that the “VRT Certified Rules for Snort v2.3” compressed file (snortrules-pr-2.3.tar.gz) is currently publicly accessible by clicking on the “Download” link next to the same title on the “Download Rules” web page through the Internet Archive’s Wayback Machine at:

https://web.archive.org/web/20060112063540/http://www.snort.org:80/pub-bin/downloads.cgi/Download/vrt_pr/snortrules-pr-2.3.tar.gz and, therefore, that this file was archived by the Internet Archive on January 12, 2006:



95. After downloading the compressed “VRT Certified Rules for Snort v2.3” file (snortrules-pr-2.3.tar.gz), I extracted the contents to a directory,

snortrules-pr-2.3, by doubleclicking on the archive. The directory listing is shown below:



96. The “doc” and “rules” folders contained therein, as illustrated in the screenshot above, indicate that files within such folders were last modified on April 5, 2005:

97. Within the “rules” folder, I opened and reviewed the “chat.rules” file using the ‘XCode’ text editor. The contents of the “chat.rules” file are presented in CS-1042. The properties of the “chat.rules” file indicate that the file was last modified on April 5, 2005. CS-1042.

98. Also contained within the “rules” folder, I opened and reviewed the “multimedia.rules” file using the ‘XCode’ program. The contents of the “multimedia.rules” file are presented in CS-1048. The properties of the “multimedia.rules” file indicate that the file was last modified on April 5, 2005. CS-1048.

99. Within the “signatures” subfolder within the “doc” folder, I opened and reviewed the “2450.txt” file. The contents of the “2450.txt” file are presented in CS-1043. The properties of the “2450.txt” file indicate that the file was last modified on April 5, 2005. CS-1043.

100. Also within the “signatures” subfolder within the “doc” folder, I opened and reviewed the “1437.txt” file. The contents of the “1437.txt” file are presented in CS-1049. The properties of the “1437.txt” file indicate that the file was last modified on April 5, 2005. CS-1049.

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

101. 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 issues¹, and was rapidly replaced with version 3 in

¹ 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

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.*, CS-1013 (“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, 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 finally [*sic*] released.”); CS-1014. 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

(TLS) clients and servers establish connections, they never negotiate the connection of Secure Sockets Layer (SSL) version 2.0.” CS-1016.

protect against CBC attacks . . .” CS-1015 pp. 4-5; See also *id.*,
p. 20.

[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.” CS-1015 p. 74.

102. TLS 1.2 was still the most up to date version of the protocol by March
2013. CS-1037.

103. As I discussed above, the well-known SSL and TLS protocols insert
an encrypted channel running on top of TCP, and underneath HTTP. CS-1038
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
an HTTP/TLS client and an HTTP/TLS server (CS-1015 p. 13, CS-1038 p. 3):

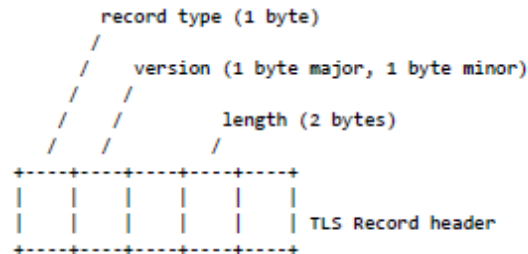
- *Handshake* subprotocol: allows the client and server to authenticate
each other and to negotiate the cipher suite and other parameters of
the connection (including SSL or TLS version). (CS-1015 pp. 29-32)

- *ChangeCipherSpec* subprotocol: makes the previously negotiated parameters effective, so communication becomes encrypted. (CS-1015 p. 24)
- *Alert* subprotocol: used for communicating exceptions and to indicate potential problems that may compromise security. (CS-1015 pp. 24-29.)
- *Application Data* subprotocol: It takes arbitrary application-layer data (*e.g.*, HTTP data) generally, and feeds it through the secure channel. (CS-1015 p. 50)

104. 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. CS-1015, 13, 16-19, 48.

105. 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:

Bytes 3-4: Length of data in the SSL/TLS Record (excluding the header itself):



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

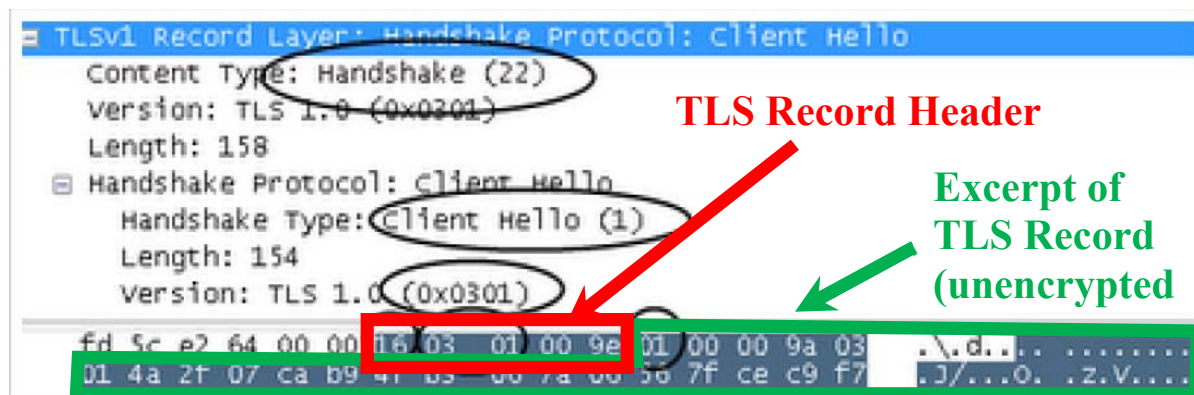
Version	Values	dec	hex
SSL 3.0		3,0	0x0300
TLS 1.0		3,1	0x0301
TLS 1.1		3,2	0x0302
TLS 1.2		3,3	0x0303

CS-1015, 16-17.

106. 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

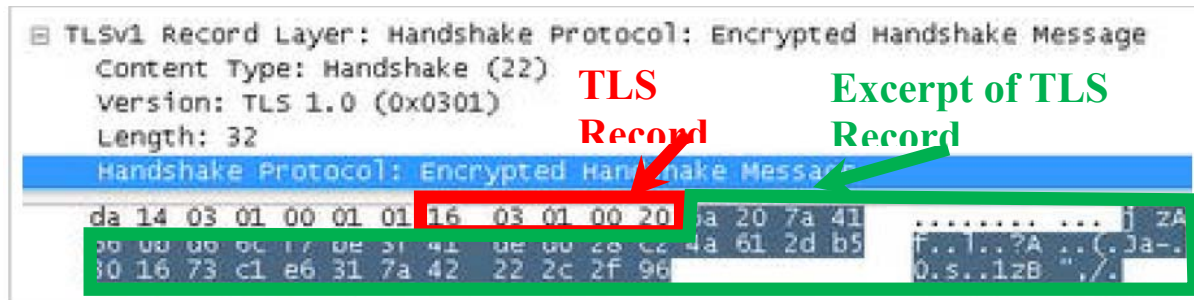
² In SSL v2 Record Protocol, the SSL Record header has no SSL version (CS-1018) and a POSA familiar with the relevant art would thus know that the version could only be obtained from looking at the unencrypted body (*i.e.* unencrypted SSL Record) of the SSL v2 handshake message packets. CS-1018.

include a TLS Record header and an unencrypted TLS Record for a “Client Hello” *handshake* subprotocol message as illustrated in the following example:



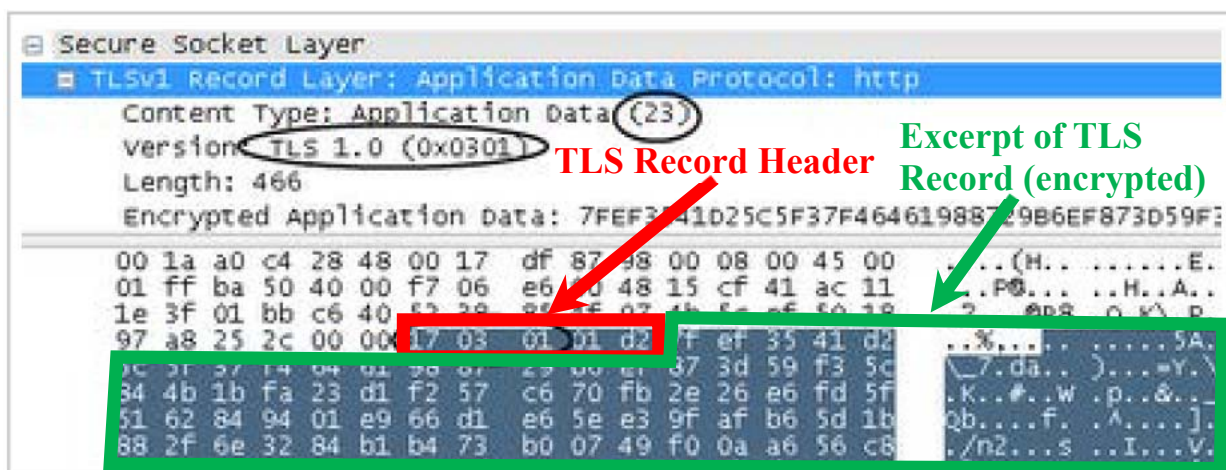
CS-1038 p. 3; CS-1039 pp. 1-2. As shown in the above example, the TLS Record header specifies that the TLS Record type is Handshake (0x16 or 22), that the TLS version is TLS 1.0 (03x01 or 3,1), and that the length of unencrypted data in the TLS Record is 154 bytes (00x9e).

107. 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:



CS-1038 p. 3; CS-1039 pp. 11-12. 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).

108. 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:



CS-1038 p. 3; CS-1039 pp. 12-13. 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).

109. As identified above, the '193 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." CS-1001 7:67-8:7.

110. As encryption of HTTP traffic became more widespread, there was a need for ways to allow enterprise IDS/IPS devices to inspect the HTTPS traffic for decryption. A well-known technique to inspect and apply rules to encrypted application layer data in HTTPS was to use an IDS or IPS (or device serving the IDS or IPS) to (1) serve as a proxy for a SSL/TLS handshake protocol between a client and a server, (2) apply packet filtering rules to the packet header data and application layer data in the handshake packets, (3) allow separate HTTPS sessions to be established between the client and proxy, and then between the proxy and server, (4) decrypt the application layer data in the HTTPS packets, and (5) inspect the application layer data and apply rules to either block or allow packets based on such rules, and then re-encrypt it for transmission to its eventual destination. See,

e.g., CS-1009 FIGS. 1, 2, 2:6-23, 3:33-37, 3:57-61, 4:25-30, 5:43-54, 5:64-6:3, 6:8-11.

111. For example, on its website³ and in its publicly accessible data sheet⁴, the Sourcefire company described a separate appliance (the “SSL Appliance”) that

³ SSL Appliances Webpage is a printout of a web page from the Sourcefire company’s publicly available website <http://www.sourcefire.com/security-technologies/cyber-security-products/3d-system/ssl-encryption-decryption>. CS-1010. I understand that the printout of the SSL Appliances Webpage at CS-1010 is currently publicly accessible through the Internet Archive’s Wayback Machine at <https://web.archive.org/web/20101220151851/http://www.sourcefire.com/security-technologies/cyber-security-products/3d-system/ssl-encryption-decryption> and was archived by the Internet Archive on December 20, 2010. CS-1010.

⁴ SSL Appliances Data Sheet (CS-1011) is a printout of the “Sourcefire SSL Appliance Fact Sheet” downloaded from a hyperlink of the same name on the NDM company’s publicly available website <http://www.ndm.net:80/sourcefirestore/Product-Data-Sheets/product-data-sheets> (“NDM Webpage”). CS-1017; CS-1011. I understand that the NDM Webpage is currently publicly accessible through the Internet Archive’s Wayback Machine at <https://web.archive.org/web/20111028104123/http://www.ndm.net:80/sourcefirestore>

a network administrator could install and use to protect a network along with, for example, a 3D Sensor with IPS (See Section X.A below) so that the IPS could inspect and apply various rules to non-SSL and decrypted SSL packets to either block or allow such packets:

[ore/Product-Data-Sheets/product-data-sheets](#) and was archived by the Internet Archive on October 28, 2011. CS-1017. I understand that the SSL Appliances Data Sheet (CS-1011) is currently publicly accessible through the Internet Archive's Wayback Machine at <https://web.archive.org/web/20120316165725/http://www.ndm.net:80/sourcefirestore/downloads/Sourcefire%20SSL%20Appliance%20Fact%20Sheet%20%28English%29.pdf> and by selecting the hyperlink "Sourcefire SSL Appliance Fact Sheet" on the NDM Webpage. CS-1017; CS-1011. I understand that the SSL Appliances Data Sheet has a copyright date of "© 2010 Sourcefire, Inc." and was archived by the Internet Archive on March 16, 2012. CS-1011, 3. I also understand that, on the NDM Webpage, the NDM company identifies itself as a "Sourcefire Authorized Distributor." CS-1017.

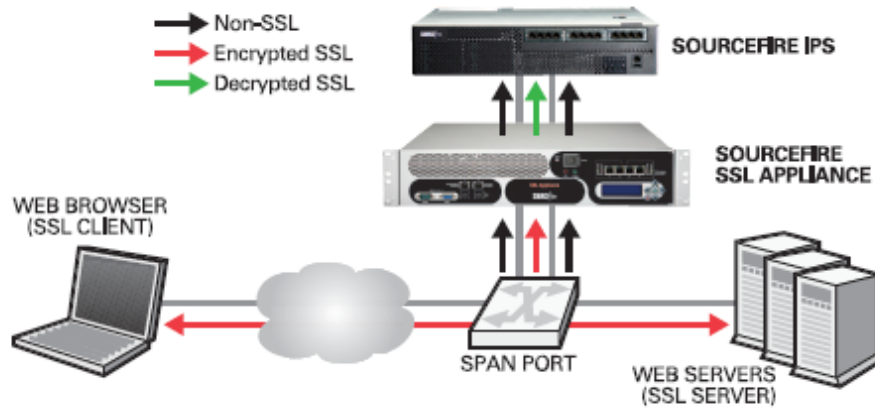


Figure 1. Passive IDS Configuration

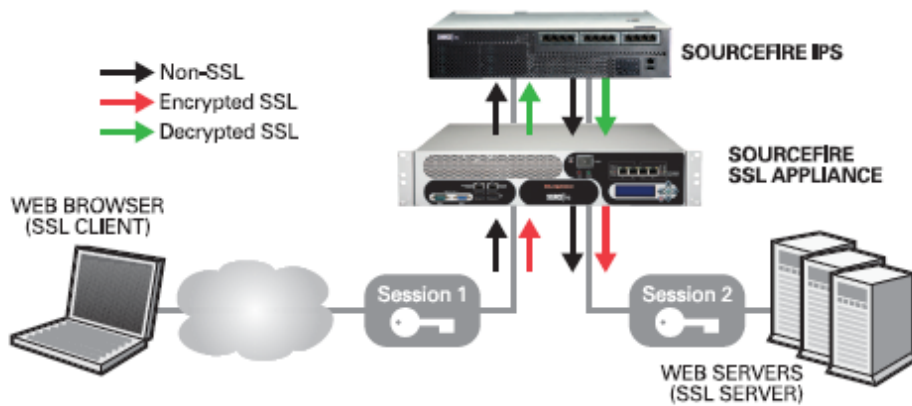


Figure 2. Inline IPS Configuration

CS-1010; CS-1011.

112. Based on the above-described historical development of the network security systems art, my opinion is that the claimed features of the '193 Patent were well-known in the art and were commonly used by many stateful firewall and IPS systems of the period between the late 1990s and late 2000s. These conventional stateful firewall and IPS systems allowed a user to customize rules, and combine any number of such rules, to customize a security policy particular to the user's specific network environment and to filter packets using (1) 5-tuple

criteria (including (i) both the source and destination IP addresses, which represented the source network from which the packets originated and the destination network to which they are directed, and (ii) the destination port in the 5-tuple which reflected the application layer protocol (*e.g.*, HTTP, FTP, SMTP, etc.) being used), and (2) based on other application-layer data.

113. In this declaration, I have analyzed the claims of the '193 Patent against a version 4.10 user guide for the Sourcefire 3D System (CS-1004), a commercial IPS derivative of the Snort open source IDS that I referenced above. This user guide (CS-1004) describes certain functionality of the Sourcefire 3D System which was very well-known to a POSA prior to March 2013 as it was widely available, was based on the leading open-source network IDS (Snort), and had a large user community developing substantial collections of rules for it. Moreover, as I outline in detail below, the user manual (CS-1004) discloses a two-stage packet filtering process that uses customizable rules to filter packets using the 5-tuple (Stage I), as well as filtering based on application layer data including PUT and GET commands, and TLS version to block or allow the packets (Stage II).

VIII. Claim Construction

114. I have been informed and understand that in order to properly evaluate the '193 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.

115. In order to construe the claims, I have reviewed the entirety of the '193 Patent (CS-1001) and its prosecution history (CS-1002). Consistent with these standards, in my opinion, no specific construction of any claim term is required because the prior art that I rely on in my Declaration meets each of the claim terms under any reasonable construction.

IX. Overview

116. In my opinion, the Challenged Claims of the '193 Patent are unpatentable over the Sourcefire 3D System User Guide Version 4.10 ("Sourcefire") in view of the knowledge and skill of a POSA as described in more detail below.

117. Sourcefire describes 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. Sourcefire describes using various rules

in such an intrusion policy to filter packets using (1) 5-tuple rule criteria (including (i) both the source and destination IP addresses, which represented the source network from which the packets originated and the destination network to which they are directed, and (ii) the destination port in the 5-tuple which reflected the application layer protocol (*e.g.*, HTTP, FTP, SMTP, etc.) being used), and (2) “keywords” based on packet content (*e.g.*, HTTP methods such as GET and PUT, FTP commands such as PORT, and encryption protocol version (*e.g.* TLS 1.0)).

118. The only difference between the Sourcefire reference and the Challenged Claims are implementation details that are well-within the knowledge and capability of a person of ordinary skill in the art. Specifically, in my opinion all the challenged claims can be implemented by simply writing appropriate Snort rules, with no modification of the underlying Sourcefire system itself required. Writing Snort rules is a task a POSA would certainly have been expected to be capable of in 2013. See also Sections IV ¶¶23-24; VI.C ¶83, VI.D. Therefore, the Challenged Claims are obvious over Sourcefire in view of the knowledge of a POSA.

X. Challenge: Claims 1-20 are Obvious under 35 U.S.C. § 103 over Sourcefire in View of the Knowledge and Skill of a POSA

A. Sourcefire Teaches Every Element of Claims 1-20 in View of the Knowledge and Skill of a POSA

119. The Sourcefire reference is a user manual titled “Sourcefire 3D System User Guide” Version 4.10 (“Sourcefire”).

120. I have reviewed the Declaration of John Leone 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. CS-1005. Based on this declaration, 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>). CS-1005.

121. I have been informed and understand that Sourcefire is a printed publication and is available as prior art under 35 U.S.C. § 102(b). Sourcefire was not cited or applied by the Examiner during the prosecution of the ’193 Patent.

1. Sourcefire

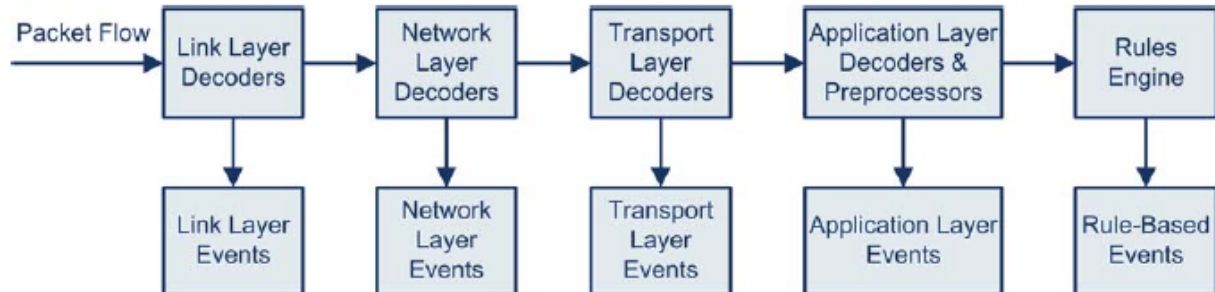
122. Sourcefire is a user manual for the Sourcefire 3D System. Sourcefire describes that the Sourcefire 3D System could identify changing assets and

vulnerabilities on the network, determine the types of attacks against the network and their impact, and defend the network in real time. CS-1004 p. 32.

123. Sourcefire describes packet-filtering devices (3D Sensors) of the 3D System that a user (*e.g.* a network security administrator) deployed and configured in his or her network to 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.” CS-1004 pp. 33-34.

124. Sourcefire describes that IPS allowed the 3D Sensor running the IPS to monitor 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. CS-1004 pp. 35, 254. 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 exploits that attackers had developed:



CS-1004 pp. 35, 254-256.

125. Sourcefire discloses that the layer-specific decoders (as illustrated above) of the IPS converted the packet headers and payloads of packets captured from the network by the 3D Sensor into a format that could easily be used by the preprocessors and rules engine. The decoders also detected a variety of possibly suspicious error or anomaly conditions in the packet headers and alerted or blocked packets based on abstract “decoder rules” that indexed these different error checks. Sourcefire discloses that the preprocessors of the IPS normalized packets at the application layer, detected protocol anomalies (such checks were indexed as preprocessor rules), and could be invoked by the rules engine of the IPS based on certain “keywords” of intrusion rules (discussed in more detail below). Sourcefire discloses that the main intrusion rules engine of the IPS implemented intrusion rules to determine whether the packet headers and/or payloads of received packets triggered one or more of such rules. CS-1004 pp. 256-259, 513, 2084, 2089.

126. Sourcefire discloses that the 3D Sensor with IPS could be deployed in an inline detection mode (rather than passive mode) to drop or replace packets that the various rules implemented on the Sensor detected as being harmful. CS-1004 pp. 262-265, 276-277, 761, 2093.

127. Sourcefire discloses that the rules supplied with a 3D Sensor with IPS were developed by the Sourcefire Vulnerability Research Team (VRT). These decoder rules, preprocessor rules, and intrusion 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 his or her network. IPS also allowed a user to create his or her own custom intrusion rules tuned to the user's specific network environment and the types of anomalies and malicious activities that such network environment is likely to encounter. CS-1004 pp. 256-260, 428-430, 761-770.

128. Sourcefire describes that, in operation, when a decoder, preprocessor, or rules engine of a 3D Sensor with IPS identified a possible intrusion, it generated an intrusion event, which was a record of the date, time, type of exploit, one or more values of the 5-tuple from the packet header of the triggering packet, whether the triggering packet was dropped or allowed, a classification of the rule that generated the event, and contextual information about the source of the attack and

its target. For packet-based events, a copy of the packet or packets that triggered the event was also recorded. CS-1004 pp. 253-260, 276-278.

129. Sourcefire describes the use of a central management point called Defense Center to manage the 3D Sensors including providing updated security policies and software updates for the 3D Sensors. CS-1004 p. 2078. In a Sourcefire 3D System deployment that included 3D Sensors with IPS and a Defense Center, the 3D Sensors transmitted events and sensor statistics to the Defense Center where the aggregated data could be viewed by a user to gain a greater understanding of the attacks against the network. CS-1004 p. 35.

130. Sourcefire discloses that the IPS allowed a user to modify various parameters of the decoder, preprocessor, and intrusion rules to customize the rules as appropriate or desired for his or her network. Sourcefire also discloses that IPS allowed a user to write his or her own custom text intrusion rules tuned to the user's specific network environment and the types of anomalies and malicious activities that such network environment is likely to encounter. Sourcefire explains that the most successful rules targeted traffic that may attempt to exploit known vulnerabilities rather than specific known exploits. By writing rules and specifying the rule's event message, the 3D System could more easily identify traffic that indicated attacks and policy evasions. CS-1004 pp. 403-406, 435-439, 485, 762-776.

131. Sourcefire describes that all of the intrusion rules had 5-tuple values associated with them. CS-1004 pp. 256-259, 761-762. Sourcefire explains that intrusion rules⁵ contained two logical parts: the rule header and the rule options.

132. The rule header part contained:

- the 5-tuple:
 - 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. CS-1004 pp. 761-770.

133. The rule options part contained:

- event messages;

⁵ Sourcefire refers to “standard text” rules and “shared object” rules as being different types of “intrusion rules.” Shared object rules include customizable “rule header” parts but not customizable “rule options” parts. CS-1004, pp. 405-406, 761-771. For purposes of this Declaration, I refer to “standard text” intrusion rules as “intrusion rules.”

- keywords and their parameters and arguments;
- patterns that a packet's payload must match to trigger the rule; and
- specifications of which parts of the packet the rules engine should inspect.

CS-1004 pp. 761-770.

134. Sourcefire provides an example intrusion rule to illustrate the different parts of the rule, *i.e.*, rule header (including parameters and arguments) and rule options (including keywords and arguments):

Rule Header

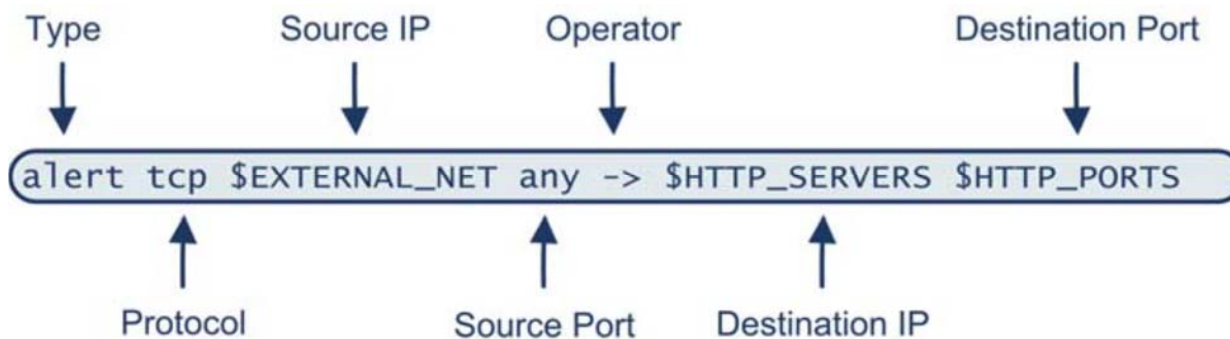
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
```

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; )
```

CS-1004 pp. 762-763.

135. Sourcefire explains that every intrusion rule had parameters and arguments that made up the rule header (including the 5-tuple). For example, the following illustrates the parts of a rule header in the example intrusion rule:



CS-1004 p. 764.

136. The Rule Header Values table below provides an example of values found in a received packet header, *e.g.*, source and destination IP addresses, and how they could be used by IPS in applying the rule header shown above:

Rule Header Values

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

CS-1004 pp. 764; 762-763.

137. Sourcefire discloses that by using the rules language, a user could specify the behavior of a rule by combining keywords in the non-rule-header part

of a rule. Keywords and their associated values (called “arguments”) in each intrusion rule dictated how the rules engine, and any relevant preprocessors invoked by the rule, tested and evaluated packets and packet-related values. The rules engine and preprocessors supported keywords in intrusion rules that allowed performance of inspection functions, such as content matching, protocol-specific pattern matching, and state-specific matching. Sourcefire discloses that the rules engine and preprocessors allowed the combination of any number of compatible keywords to create highly specific intrusion rules. Sourcefire also discloses that the rules engine and preprocessors allowed the combination of any number of rules to customize a security policy tuned to the user’s specific network environment. This helped decrease the chance of false positives and false negatives and focused the intrusion-based information that a user received. CS-1004 pp. 769-770, 333-335.

138. Like the claims of the ’193 Patent (See Sections VI.A, VI.C), Sourcefire discloses that, for a given 3D Sensor with IPS, multiple intrusion rules could be used to filter packets. These rules used the 5-tuple of the packet header to filter the packets. This packet header data included both the source and destination IP addresses, which represent the source network from which the packets originated (*i.e.*, the “first network” of claim 1) and the destination network to which they are directed (*i.e.*, either the “second network” or the “third network” of

claim 1). The destination port⁶ in the 5-tuple also reflects the application layer protocol (*e.g.*, HTTP, FTP, SMTP, etc.) being used for the data transfer. CS-1004 pp. 256-259, 764-769.

139. 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. CS-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.

140. 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,

⁶ A POSA understood that the destination port in client → server communication packets is the source port in server → client communication packets.

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. CS-1004 pp. 598, 697-701.

141. 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 message⁷ packets for an encrypted session:

ssl_version Arguments

Argument	Purpose
ssl_v2	Matches against traffic encoded using Secure Sockets Layer (SSL) Version 2.
ssl_v3	Matches against traffic encoded using Secure Sockets Layer (SSL) Version 3.

ssl_version Arguments (Continued)

Argument	Purpose
tls1.0	Matches against traffic encoded using Transport Layer Security (TLS) Version 1.0.
tls1.1	Matches against traffic encoded using Transport Layer Security (TLS) Version 1.1.
tls1.2	Matches against traffic encoded using Transport Layer Security (TLS) Version 1.2.

CS-1004 pp. 827-828, 491, 597-601, 700.

142. When an intrusion rule used the `ssl_version` keyword, including in the “adaptive” mode, the rules engine invoked the SSL preprocessor to parse SSL/TLS traffic, find SSL/TLS version information, and either *pass* (allow), *alert* (and pass),

⁷ As I discussed above, the SSL/TLS Record of “Finished” handshake message packets is encrypted. Section VII.E, ¶¶103-107.

or *drop* (block) packets identifying the specified version. CS-1004 pp. 827-828, 491, 597-601, 435-439, 697-701. As I discussed above, in the SSL v2 Record Protocol, the SSL Record header has no version and a POSA familiar with the relevant art would thus know that the version could only be obtained from looking at the unencrypted body of the SSL v2 handshake message packets. Section VII.E, ¶105; CS-1018. However, for higher versions of SSL/TLS, it would be obvious to a POSA that the SSL preprocessor must look at all SSL/TLS Record headers (in order to find the SSL/TLS Record boundaries and extract the information in the SSL/TLS Record headers). See also Section VII.E, ¶¶103-108. Thus, a POSA would understand that the Sourcefire-described SSL preprocessor could easily find SSL/TLS version information from either the unencrypted SSL/TLS Records of handshake message packets, or the SSL/TLS Record headers of SSL/TLS Record Protocol packets (*e.g.*, handshake message packets, application-data message packets), or both.

143. 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. CS-1004 pp. 785-786, 700.

144. As another example, Sourcefire discloses that FTP content keywords of intrusion rules, including in the “adaptive” mode, could be used by another application-layer preprocessor called the FTP/Telnet preprocessor and rules engine of a 3D Sensor to search for matched application layer content in packets such as known FTP commands (*e.g.*, “PORT”) or known “anonymous” usernames, and either *pass* (*i.e.*, allow), generate an *alert* (and pass), or *drop* (*i.e.*, block) packets identifying such an FTP command in their application layer payload content. CS-1004 pp. 556, 765, 538-542, 546-560, 700; CS-1040 pp. 2-3.

145. As a further example, Sourcefire discloses that instant message 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. CS-1004 pp. 266, 334 (“Your internal security policies guide how you tune the decoder engine, preprocessor engine, and rules engine. For example, if your security policies prohibit instant messaging, you may want to identify instant message traffic traversing your network.”), 333, 335, 373 (identifying the predefined \$AIM_SERVERS rule variable which could be used in “chat-based

rules and rules that look for [AOL Instant Messenger] exploits.”), 428 (identifying the “Chat” rule category which includes “[r]ules that detect the use of unauthorized instant messaging programs.”), 874, 2032-2043, 2092; Section VII.D, ¶88.

146. 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. CS-1004 pp. 266, 333-335, 429-430, 874, 1453, 1470-1471, 2034, 2037, 2039, 2040-2042, 2092; Section VII.D, ¶89.

147. 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.

B. Detailed Invalidity Analysis

148. As detailed below, in my opinion, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses every element of Claims 1-20.

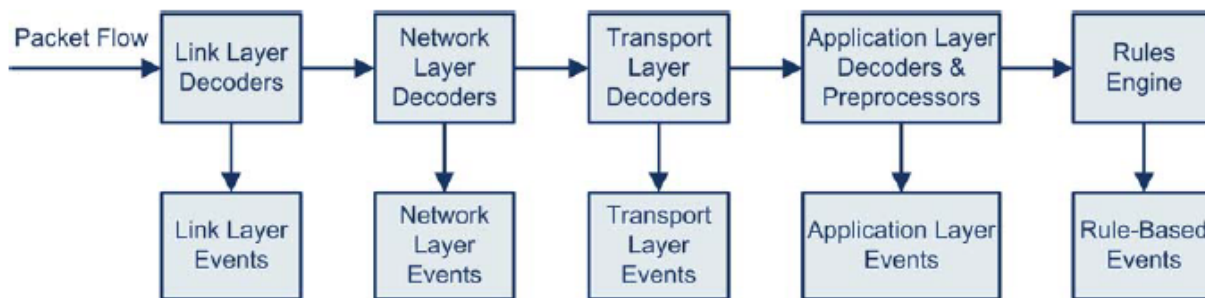
Each of the limitations of these claims are disclosed by Sourcefire and any potential differences that may be argued by the Patent Owner are implementation details that are well-within the knowledge and capability of a POSA (for example only requiring creating new Snort rules for well-documented protocols).

Claim 1: *[1.0]⁸ A method comprising: receiving, by a computing system and from a computing device located in a first network, a plurality of packets, wherein the plurality of packets comprises a first portion of packets and a second portion of packets;*

149. Sourcefire discloses this limitation.

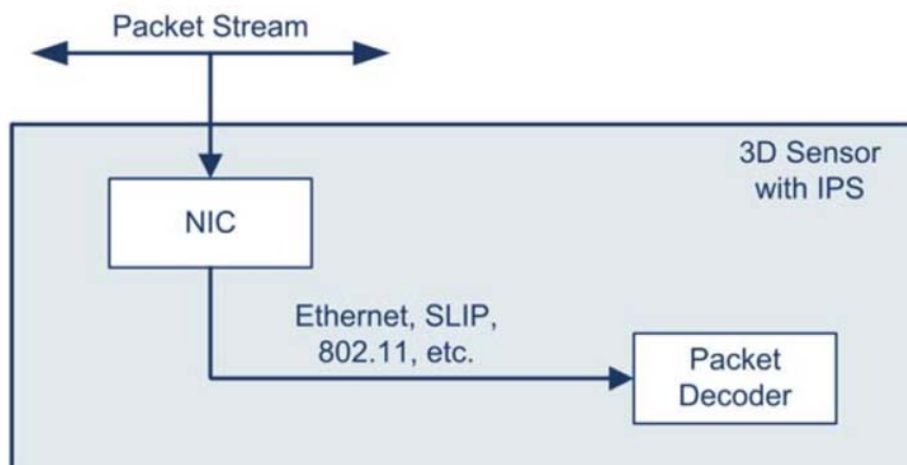
150. Sourcefire describes that 3D Sensors with Intrusion Prevention System (IPS) monitored a network for attacks that might affect the availability, integrity, or confidentiality of hosts located in a network that it was designed to protect. Each deployed 3D Sensor with IPS examined the packets that traversed the monitored 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 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:

⁸ Throughout this Declaration, the citation to claim elements is to [claim.element].



CS-1004 p. 35, 253-256.

151. Sourcefire discloses that each 3D Sensor received packets through a communication interface. CS-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:



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

152. Thus, Sourcefire discloses 3D Sensors receiving a plurality of packets, some of which can be deemed to be a first portion of packets and some of which can be deemed a second portion of packets, and the 3D Sensor is provisioned with a plurality of packet-filtering rules. Sourcefire discloses that the packets received by the 3D Sensor originate from a computing device (*e.g.*, host) located in a first network where the first network could be the network that the 3D Sensor is designed to protect (*e.g.*, HOME_NET) or a network external to such network (*e.g.*, EXTERNAL_NET). CS-1004 pp. 213-214, 225-227, 234-235, 260, 277-278, 333-335, 373-379.

153. Therefore, Sourcefire discloses this limitation.

[1.1] responsive to a determination by the computing system that the first portion of packets comprises data corresponding to criteria specified by one or more packet-filtering rules configured to prevent a particular type of data transfer from the first network to a second network, wherein the data indicates that the first portion of packets is destined for the second network:

154. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

155. As I discussed above, Sourcefire discloses that IPS allowed a user to modify various parameters of the decoder, preprocessor, and intrusion rules to customize the rules as appropriate or desired for his or her network. Sourcefire also discloses that IPS allowed a user to write his or her own custom intrusion rules tuned to the user's specific network environment and the types of anomalies and

malicious activities that such network environment is likely to encounter.

Sourcefire explains that the most successful rules targeted traffic that may attempt to exploit known vulnerabilities rather than specific known exploits. By writing rules and specifying the rule's event message, the system could more easily identify traffic that indicated attacks and policy evasions. CS-1004 pp. 403-406, 435-439, 485, 762-770.

156. Sourcefire describes that all of the intrusion rules had 5-tuple criteria associated with them. CS-1004 pp. 256-259, 761-770. The 5-tuple criteria included the source IP addresses and port and the destination IP addresses and port. CS-1004 pp. 358, 488-490, 761-762. Sourcefire describes that the IP address 5-tuple rule criteria could be a range of IP addresses. CS-1004 pp. 762-763, 767, 769, 702-704, 59-60. A POSA understood that this 5-tuple rule criteria could be used to filter packets transferring data from a first network (source) to a second network (destination), for example, from an internal protected network to an external network destination and vice versa. For example, the 5-tuple for each packet in the first portion of packets would include data that indicated the first network contained the source IP address and the second network contained the destination IP address. CS-1004 pp. 358, 488-490, 761-770.

157. The well-known destination port in the 5-tuple of these packets, in the usual case, also reflected the application layer protocol (*e.g.*, HTTP (port 80),

HTTPS (port 443), FTP (port 21), SMTP (port 25), XMPP (port 5222 or 5269), etc.) being used for the data transfer if such packets were traveling over a standard port. CS-1004 pp. 256-259, 764-769, 54-55; CS-1044, pp. 43-44; CS-1050 pp. 24-25.

158. As I discussed above, Sourcefire also discloses that by using the rules language, a user could specify the behavior of a rule by combining keywords in the rule. Keywords and their associated values (called “arguments”) in each intrusion rule dictated how the rules engine, and any relevant preprocessors invoked by the rule, tested and evaluated packets and packet-related values. The rules engine and preprocessors supported keywords in intrusion rules that allowed performance of inspection functions, such as content matching, protocol-specific pattern matching, and state-specific matching. Sourcefire discloses that the rules engine and preprocessors allowed the combination of any number of compatible keywords to create highly specific intrusion rules. Sourcefire also discloses that the rules engine and preprocessors allowed the user to combine any number of rules to customize a security policy tuned to the user’s specific network environment. This helped decrease the chance of false positives and false negatives and focused the intrusion information that a user received. CS-1004 pp. 769-770.

159. For example, Sourcefire provides guidelines for creating rules to implement a security policy including “understand[ing] the traffic that traverses the

network”, “set[ing] your variables to accurately reflect your home and external networks” and including guidance that “[i]f none of the existing intrusion rules meet your needs, write new rules that inspect for intrusion attempts.” CS-1004 pp. 333-335. Thus, Sourcefire expressly teaches a POSA to customize/tailor rules and rule combinations for whatever purpose is set forth in the security policy and provides the motivation for a POSA to implement the specific rules and rule combinations recited in the Challenged Claims.

160. As I discussed above, Sourcefire describes that, in addition to the destination port (and other) 5-tuple rule criteria, keywords of intrusion rules could be used to search for matched application-layer content in packets such as, for example, data associated with particular types of data transfer (*e.g.*, HTTP, FTP, SMTP, IM, multimedia, HTTP methods, FTP commands, HTTP over SSL/TLS (HTTPS), TLS-version, etc.). CS-1004 pp. 827-828, 491, 597-601, 435-439, 785-786, 334, 373, 428-429, 874, 2032-2043, 2092. Sourcefire describes the type of actions that could be triggered by the rules as *pass* (and ignore), *alert* (and pass) or *drop* (block). CS-1004 pp. 762, 764-769, 358-359. Thus, a POSA understood that the application of various rules could cause a first portion of packets from the first network addressed to the second network to be dropped.

161. For example, Sourcefire discloses that, in addition to the 5-tuple criteria of destination port 80, rule keywords could be used by the HTTP inspect

preprocessor and rules engine of a 3D Sensor to filter received packets based on rule criteria including the most common HTTP methods of GET, PUT, POST and CONNECT. CS-1004 pp. 560, 568, 786, 700. A POSA understood that identifying HTTP methods was important in existing packet filtering firewalls with stateful inspection because application methods were used as part of rule criteria to block and pass packets, *e.g.*, a known “stateful inspection firewall allowed the FTP ‘GET’ command but disallowed the ‘PUT’ command” (CS-1008 4:31-41), vulnerabilities with certain HTTP methods (*e.g.*, PUT, POST, DELETE) were well-known, and HTTP methods were used in IDS rules at least as early as 1997. CS-1028 p. 189; CS-1004 pp. 785-786, 807, 435-439, 491; Section VI.C, ¶¶77, 80, 82.

162. Therefore, a POSA understood Sourcefire disclosed how a user would have written a rule using the HTTP Method option of the HTTP content keyword as part of the rule criteria to invoke the HTTP inspect preprocessor to identify a first portion of packets using the particular “PUT” (or “POST”, “DELETE”, “CONNECT”, etc.) HTTP method type of data transfer and to block such packets with certain application payload content posing a threat from continuing towards their second network destination. Moreover, a POSA would have been motivated to write an intrusion rule to do this and have the IPS apply such a rule such as, for example, in the “adaptive” mode to identify and block packets associated with this

particular type of data transfer (HTTP “PUT” method) and not traveling over standard TCP port 80. CS-1004 pp. 697-701; Section VII.C, ¶¶77, 80, 82.

163. As another example, Sourcefire discloses that, in addition to the 5-tuple criteria of destination port 21 (FTP), rule keywords could be used by the rules engine of a 3D Sensor to filter received packets based on rule criteria including FTP command (*e.g.*, the known “PORT” FTP command). CS-1004 pp. 556, 765, 538-542, 546-560, 700, 492, 776-783, 861-864; Section X.A, ¶144. As described in Sourcefire, and as was known to a POSA, identifying this particular type of data transfer was important to address the well-known Bounce attack. CS-1004 p. 556; CS-1040 pp. 2-3; Section X.A, ¶144.

164. Therefore, a POSA understood Sourcefire disclosed how a user would have written a rule using content keywords as part of the rule criteria to identify a first portion of packets using the particular “PORT” FTP command type of data transfer and to block such packets from continuing towards their second network destination. Moreover, a POSA would have been motivated to write an intrusion rule to do this and have the IPS apply such a rule such as, for example, in the “adaptive” mode to identify and block packets associated with this particular type of data transfer (FTP “PORT” command) and not traveling over standard TCP port 21. CS-1004 pp. 697-701; Section X.A, ¶144.

165. As a further example, Sourcefire discloses that, in addition to the 5-tuple criteria of destination port 443 (HTTPS), rule keywords could be used by the SSL preprocessor (when invoked by the rules engine) of a 3D Sensor to filter received packets based on rule criteria including encryption protocol version (*e.g.* TLS or SSL version). Sourcefire discloses that that the keyword `ssl_version` could be included in such intrusion rules and used to identify and block a first portion of packets (*e.g.*, vulnerable SSL/TLS Record Protocol packets) from continuing towards their second network destination. CS-1004 pp. 827-828, 491, 597-601, 435-439. A POSA understood that certain SSL and TLS versions contained vulnerabilities (*e.g.*, TLS 1.0, SSL v2, SSL v3) and Sourcefire teaches that a *drop* rule could be written to block packets based on a `ssl_version` that included a known vulnerability. CS-1037; CS-1038; CS-1039 p. 4; CS-1013; CS-1014; CS-1015.

166. Because security vulnerabilities associated with HTTPS data transfers using or proposing to use TLS 1.0, SSL v2, SSL v3 were widely known, a POSA would be motivated to write appropriate *drop* rules to identify and block SSL/TLS Record Protocol packets that proposed using (or did use) these particular less secure SSL and TLS types of data transfer from continuing toward their second network destination. Moreover, a POSA would have been motivated to write an intrusion rule to do this and have the IPS apply such a rule, for example, in the “adaptive” mode to identify and block SSL/TLS Record Protocol packets

associated with these particular less secure SSL and TLS types of data transfer and not traveling over standard TCP port 443. CS-1004 pp. 697-701; Section VII.E, ¶101.

167. As a further example, a POSA reading Sourcefire understood that multimedia 5-tuple and content keyword rule criteria could be used by the rules engine to identify matched content in a first portion of packets and block such packets from continuing towards their second network destination, for example, to enforce a security policy prohibiting employees' use of such applications. A POSA might use the default UDP port 5004 for transferring audio/video content to block some RTP traffic, or use fixed RTP header fields as content keyword targets or block SIP/SDP packets in a control connection that indicated the use of RTP (*e.g.* with content string 'RTP/AVP' or 'a=rtpmap'). CS-1004 pp. 266, 333-335, 429-430, 1453, 1470-1471, 2030, 2034, 2037, 2039, 2040-2042, 2092; CS-1050 pp. 24-25; CS-1050 pp. 77-79; CS-1052 p. 7; Section VII.D, ¶89.

168. A POSA understood that such security policies in enterprise networks restricted employees' use of high bandwidth applications like streaming media while also avoiding known vulnerabilities posed by such use. *Id.*; Section VII.D, ¶89. Thus a POSA understood that enforcing security policies which disallowed the use of known protocols for transferring streaming multimedia (including RTP) on an internal network was simply a design choice to yield predictable results of

reducing employee bandwidth usage with increased security on such network because doing so would involve nothing more than applying a known technique to a known device ready for improvement.

169. As I discussed above, and as acknowledged by the '193 Patent, there is no technological difference between using particular type of data transfer criteria (*e.g.*, HTTP (or HTTP method), FTP (or FTP command), multimedia data (*e.g.*, RTP transfer)), over other types of rule criteria (*e.g.*, SSL/TLS version value, messaging protocol (*e.g.*, AIM, Yahoo! Messenger, XMPP)), and there are no specific technology changes needed to accommodate one data transfer protocol (*e.g.*, HTTP data transfer) over another (*e.g.*, messaging, encryption, FTP, RTP), or one data transfer method (*e.g.*, HTTP GET, FTP GET) over another (*e.g.*, HTTP PUT, FTP PORT). See also Section VI.C.

170. Moreover, as I also discussed above and as also acknowledged by the '193 Patent, there are no specific technology changes needed to accommodate allowing or blocking packets specifying one destination and origin from packets specifying any other possible destination and origin. See also Section VI.C. Thus, it is simply a design choice to select from a finite number of identified, rule criteria that are relevant to a selected network vulnerability (*e.g.*, HTTP PUT method or FTP PORT command, and certain application payload content (*e.g.*, IM content, multimedia content)) to yield predictable results of increased network security

because doing so would involve nothing more than applying a known technique to a known device ready for improvement.

171. Thus, Sourcefire teaches that various packet-filtering rules could be configured to prevent a particular type of data transfer from the first network to a second network, wherein the data satisfies rule criteria based on HTTP method, FTP command, IM content, multimedia content (*e.g.*, RTP streams), TLS_version, or any other application layer criteria designed to prevent data transfer subject to a known vulnerability or exploit. CS-1004 pp. 333-335, 435-439, 491-492, 538-542, 546-560, 597-601, 697-701, 765, 785-786, 807, 827-828.

172. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[1.2] 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; and

173. Sourcefire discloses this limitation.

174. Using the HTTP method drop rule discussed above as an example, a drop operator would be applied to the first portion of packets that include an HTTP method of PUT. See also Claim [1.1].

175. Likewise using the FTP command drop rule discussed above as an example, a drop operator would be applied to the first portion of packets that include an FTP command of PORT. See also Claim [1.1].

176. Likewise using the ssl_version drop rule discussed above as an example, a drop operator would be applied to the first portion of packets having a version of TLS 1.0, SSL v2, or SSL v3. See also Claim [1.1].

177. Therefore, Sourcefire discloses this limitation.

[1.3] dropping, by the computing system, each packet in first portion of packets; and

178. Sourcefire discloses this limitation.

179. Applying the HTTP method drop rule, FTP command drop rule, or TLS version drop rule from the examples above will result in dropping each packet in the first portion of packets. See also Claim [1.1].

180. Therefore, Sourcefire discloses this limitation.

[1.4] responsive to a determination by the computing system that the second portion of packets comprises data that does not correspond to the criteria wherein the data indicates that the second portion of packets is destined for a third network:

181. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

182. As I discussed above, Sourcefire discloses 3D Sensors receiving a plurality of packets, some of which can be deemed to be a first portion of packets

from a first network (source) and destined for a second network (destination) where the criteria are associated with a particular type of data transfer such as, for example, one of HTTP method (port 80 or non-standard ports), FTP command (port 21 or non-standard ports), or ssl_version (port 443 or non-standard ports).

183. Likewise, Sourcefire discloses that different intrusion rules could be applied to some different portion of the plurality of packets which can be deemed to be a second portion of packets from the first network (source) and destined for a third network (destination different than second network destination).

184. For example, the rule applied to drop the first portion of packets could include: (i) 5-tuple criteria specifying first network source IP addresses and source port and second network FTP destination IP addresses and FTP destination port (*e.g.*, 21, second network IP addresses), and (ii) keyword criteria specifying a FTP PORT type of data transfer. See Claim [1.1]. In this example, the different rule applied to *allow* the second portion of packets could include (i) 5-tuple criteria specifying first network source IP addresses and third network destination IP addresses and *non*-FTP destination port (*e.g.*, *not* TCP port 21), and/or (ii) keyword criteria specifying application-layer data that is *not* associated with an FTP PORT type of data transfer (*e.g.*, HTTP data, multimedia (*e.g.*, RTP) data, SSL/TLS Record Protocol data, IM data, SMTP data, etc.). The 5-tuple criteria in

the different rule could specify, for example, ANY, or alternatively a netmask for a specifically known safe/trusted network, for destination IP addresses.

185. Sourcefire discloses that the rules engine and preprocessors allowed the combination of any number of rules to customize a security policy tuned to the user's specific network environment, and Sourcefire provided guidelines on how to write rules to implement a security policy. CS-1004 pp. 333-335. In other words, Sourcefire provides a very flexible approach to crafting rules that recognized that the same rule criteria can be utilized differently for packets to/from an internal network as compared to an external network, for traffic received from or destined for a specific port, and for whether it is desired to pass, drop or alert in response to triggering a rule. CS-1004 pp. 333-335.

186. As I discussed above, and as acknowledged by the '193 Patent, there is no technological difference between combining rules that specify one particular type of data transfer criteria (*e.g.*, FTP PORT) with rules that don't specify such criteria, and other combinations that include rules that specify a different type of data transfer criteria (*e.g.*, HTTP, RTP, ssl_version), and there are no specific technology changes needed to accommodate one rule combination over another, or one packet destination and origin from another. See Claim [1.1]. Rather, it is simply a design choice to select from a finite number of rules that are each relevant to a selected network vulnerability (*e.g.*, FTP Bounce attack), and combinations of

such rules that allow for packets destined for one network (e.g., an external FTP server destination) to be treated differently than a packet destined for a different network (e.g., an external HTTP, RTP, IM, HTTPS, etc. server destination). See Claim [1.1]. In other words, Sourcefire encourages a POSA to improve IPS processing efficiency or throughput to yield predictable results tuned to improved network security of the user's particular network environment because doing so would involve nothing more than applying a known technique to a known device ready for improvement.

187. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[1.5] applying, by the computing system and to each packet in the second portion of packets, and without applying the one or more packet-filtering rules configured to prevent the particular type of data transfer from the first network to the second network, a second operator configured to forward packets not associated with the particular type of data transfer toward the third network; and

188. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

189. Sourcefire discloses that the 3D System with IPS can apply different *pass* (or *alert* (and *pass*) rules to packets (e.g., HTTP, SMTP, RTP, IM, SSL/TLS Record Protocol packets) that are not associated with, for example, FTP packets including application-layer data specifying a FTP PORT command. See Claim [1.1]. In other words, a POSA would understand that, as a matter of design choice,

a user could also implement intrusion rules or preprocessor rules which include a forward operator (*e.g.*, *pass*, *alert* (and *pass*)) configured to have the applicable preprocessor (*e.g.*, HTTP Inspect, SSL), and/or rules engine, forward *non*-FTP packets (*e.g.*, HTTP, SSL/TLS Record Protocol, SMTP, RTP, IM packets) toward their third network destination IP address. See Claim [1.1], [1.4].

190. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[1.6] forwarding, by the computing system, each packet in the second portion of packets toward the third network.

191. Sourcefire discloses this limitation.

192. Because, for example, the FTP PORT command rule is not applicable to the second portion of packets as I described above, and because packet data in the second portion of packets satisfies different rule criteria of other *pass* or *alert* (and *pass*) intrusion rules or *non*-FTP/Telnet preprocessor rules, each packet in the second portion of packets will be forwarded toward their third network destination IP address.

193. Therefore, Sourcefire discloses this limitation.

Claim 2: [2.1] The method of claim 1, wherein the first portion of packets comprises data indicating: a protocol type associated with the particular type of data transfer, and corresponding to the criteria specified by the one or more packet-filtering rules; and

194. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

195. As I discussed above for Claim 1, the first portion of packets comprises data (e.g., 5-tuple destination port (21), FTP PORT command, etc.) indicating a protocol type associated with the particular type of data transfer (e.g., FTP) and second network destination and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple and keyword rule criteria). See Claim [1.1].

196. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[2.2] the second portion of packets comprises data indicating a protocol type not associated with the particular type of data transfer.

197. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

198. As I discussed above for Claim 1, the second portion of packets comprises data (e.g., 5-tuple destination port (80, 443, 25), HTTP GET method, SSL/TLS version, SMTP data, multimedia (e.g., RTP) data, IM (e.g., Yahoo! Messenger, XMPP) data, etc.) indicating a protocol type not associated with the particular type of data transfer (e.g., HTTP, SSL/TLS Record Protocol, SMTP, RTP, XMPP are not associated with FTP or FTP PORT). See Claim [1.4]-[1.5].

199. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 3: [3.1] The method of claim 1, wherein: the first portion of packets comprises data indicating a first destination port number associated with the particular type of data transfer, and corresponding to the criteria specified by the one or more packet-filtering rules; and [3.2] the second portion of packets comprises data indicating a second destination port number not associated with the particular type of data transfer.

200. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses these limitations.

201. As I discussed above for Claims 1 and 2, the first portion of packets could comprise data indicating a first destination port number (e.g., 21) associated with the particular type of data transfer (e.g., FTP protocol, FTP PORT command) and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple (e.g., port 21, known FTP server IP addresses, etc.) and keyword rule criteria). See Claims [1.1], [2.1]. Likewise, the second portion of packets comprises data indicating a second destination port number (e.g., port 80, 443, 25) *not* associated with the particular type of data transfer (e.g., HTTP, SSL/TLS, SMTP). See Claims [1.4]-[1.5], [2.2] .

202. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses these limitations.

Claim 4: [4.1] The method of claim 1, wherein: the first portion of packets comprises data associated with hypertext transfer protocol (HTTP), and

corresponding to the criteria specified by the one or more packet-filtering rules; and [4.2] the second portion of packets does not comprise data associated with HTTP.

203. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

204. As I discussed above for Claim 1, the first portion of packets could comprise data associated with HTTP (e.g., port 80, HTTP PUT), and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple (e.g., port 80, known HTTP server IP addresses, etc.) and HTTP Method keyword rule criteria). See Claim [1.1]-[1.3]. Likewise, the second portion of packets does not comprise data associated with HTTP (e.g., port 21, FTP data). See Claim [1.4]-[1.5].

205. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 5: [5.1] The method of claim 1, wherein: the first portion of packets comprises data associated with hypertext transfer protocol secure (HTTPS), and corresponding to the criteria specified by the one or more packet-filtering rules; and [5.2] the second portion of packets does not comprise data associated with HTTPS.

206. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

207. As I discussed above for Claim 1, a POSA would understand the first portion of packets could be SSL/TLS Record Protocol packets comprising data

associated with HTTPS (e.g., port 443, SSL/TLS version), and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple (e.g., port 443, known HTTPS server IP addresses, etc.) and SSL/TLS version keyword rule criteria). See Claim [1.1]-[1.3]. Likewise, the second portion of packets do not comprise data associated with HTTPS (e.g., port 21, FTP data). See Claim [1.4]-[1.5].

208. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 6: *[6.1] The method of claim 1, wherein: the first portion of packets comprises data associated with file transfer protocol (FTP), and corresponding to the criteria specified by the one or more packet-filtering rules; [6.2] and the second portion of packets does not comprise data associated with FTP.*

209. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

210. As I discussed above for Claims 1-3, the first portion of packets could comprise data associated with FTP (e.g., port 21, FTP PORT command) and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple (e.g., port 21, known FTP server IP addresses, etc.) and FTP command content keyword rule criteria designed to detect a particular FTP command). See Claims [1.1]-[1.3], [2.1], [3.1]. Likewise, the second portion of

packets does not comprise data associated with FTP (e.g., port 80, 443, HTTP GET, SSL/TLS version). See Claims [1.4]-[1.5], [2.2], [3.2].

211. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 7: [7.1] The method of claim 1, wherein: the first portion of packets comprises data associated with real-time transport protocol (RTP), and corresponding to the criteria specified by the one or more packet-filtering rules; and [7.2] the second portion of packets does not comprise data associated with RTP.

212. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

213. As I discussed above for Claim 1, a POSA would understand the first portion of packets could comprise data associated with RTP (e.g., default UDP port for transferring audio/video content, fixed RTP header fields, control messages in a SIP connection setting up an RTP stream) and corresponding to the criteria specified by the one or more packet-filtering rules (e.g., 5-tuple (e.g., known RTP server IP addresses) and/or content keyword rule criteria designed to detect RTP data transfer). See Claim [1.1].

214. Indeed, the '193 Patent presumes that a POSA would know how to identify packets comprising data associated with RTP, as the totality of its disclosure with respect to RTP is "it may be useful for a packet filter to rapidly detect if an IP packet contains a Real-time Transport Protocol (RTP) application

packet used to deliver audio or video information (e.g., if a cyber attack based on RTP has yet to be discovered, network administrators may choose to not process RTP packets through the cyber security defense systems that may be protecting their networks).” CS-1001 9:3-10; Sections VI.A, VI.C.

215. Likewise, the second portion of packets does not comprise data associated with multimedia transfer via RTP (e.g., port 21, 25, FTP data, SMTP data). See Claim [1.4]-[1.5].

216. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 8: [8.0] A method comprising: receiving, by a computing system and from a computing device located in a first network, a plurality of packets;

217. Sourcefire discloses this limitation.

218. This limitation is disclosed in Sourcefire as I discussed with respect to Claim 1. See Claim [1.0].

219. Therefore, Sourcefire discloses this limitation.

[8.1] responsive to a determination by the computing system that a first packet of the plurality of packets comprises data associated with eXtensible messaging and presence protocol (XMPP) and the data corresponds to criteria specified by one or more packet-filtering rules configured to prevent a particular type of data transfer from the first network to a second network;

220. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

221. As I discussed above for Claims 1 and 2, Sourcefire discloses packet-filtering rules which use protocol type as the criteria to decide whether to pass or drop packets to prevent them from transferring from a first network to a second network. See Claims [1.1], [2.1]. Sourcefire discloses several different data transfer protocols including HTTP, RTP, SMTP, FTP, and several different data messaging protocols (e.g., IM protocols including AIM and Yahoo! Messenger). See Claim [1.1]; CS-1004 pp. 334, 373, 398, 428, 2030-2037, 2092.

222. Furthermore, Sourcefire expressly discloses that known instant message (IM) 5-tuple rule criteria (e.g., \$AIM_SERVERS) and content keywords could be used by the rules engine of a 3D Sensor to search for matched data in packets, for example, to enforce a security policy prohibiting employees' use of such applications:

Your internal security policies guide how you tune the decoder engine, preprocessor engine, and rules engine. For example, *if your security policies prohibit instant messaging, you may want to identify instant message traffic traversing your network.*

CS-1004 pp. 334, 333, 335, 373, 428, 874, 2092.

223. While Sourcefire does not expressly identify the messaging protocol XMPP, as I discussed above Sourcefire expressly identifies other IM protocols and a POSA understood that XMPP (also referred to as Jabber) was the kind of well-known IM protocol (CS-1044; CS-1045; CS-1030) that was being discussed in

Sourcefire as a security concern. A POSA further understood that the '193 Patent treats all application-layer rule criteria the same and identifies *the exact same system architecture and method* for allowing or blocking received packets using (i) data *transfer* protocols (*e.g.*, HTTP, FTP, RTP), (ii) data *messaging* protocols (*e.g.*, XMPP) and (iii) data *encryption* protocols (*e.g.*, TLS 1.0). See also Section VI.C.

224. Indeed, the '193 Patent presumes that a POSA would know how to identify packets comprising data associated with messaging protocols such as XMPP, as the totality of its disclosure with respect to XMPP is that “these methods may provide network administrators with capabilities to prevent exfiltrations that are mediated by other data transfer protocols besides HTTP and HTTPS. Examples of such protocols include ... messaging protocols such as eXtensible Messaging and Presence Protocol (XMPP).” CS-1001 8:59-65; Sections VI.A, VI.C.

225. Thus, a POSA understood that the teaching of Sourcefire does not distinguish between any of the IM protocol types and thus is applicable to any IM protocols including XMPP. Further, a POSA understood the first portion of packets could comprise data indicating a default XMPP destination port number (*e.g.*, 5222, 5269) and/or other data (*e.g.*, XML stream tags, JabberID, “xmnl=”, “jabber: client”, etc.) associated with XMPP and corresponding to the criteria specified by the one or more packet-filtering rules (*e.g.*, XMPP 5-tuple and

keyword rule criteria). See Claim [1.1]; CS-1044 pp. 3-6, 20-22, 43-44; CS-1045 pp. 3-9, 60-62.

226. Moreover, vulnerabilities with IM protocols generally (Section VII.D, ¶88; CS-1004 pp. 334, 428, 1471. 2092), and specifically with XMPP (CS-1044 pp. 41-44; CS-1045 p. 58; CS-1047), were known to a POSA, and thus a POSA understood that enforcing security policies which disallowed employees' use of all known IM protocols, including XMPP from an internal network to a second network (*e.g.*, an external network) was simply a design choice to yield predictable results of increased security on such network because doing so would involve nothing more than applying a known technique to a known device ready for improvement. See Claim [1.1].

227. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[8.2] applying, by the computing system and to the first packet, a first operator, specified by the one or more packet-filtering rules, configured to drop packets associated with the particular type of data transfer; and

228. Sourcefire discloses this limitation.

229. For the reasons I discussed regarding Claim 1, Sourcefire discloses that a *drop* operator would be applied to the first packet that meets the XMPP criteria. See Claim [1.1]-[1.3].

230. Therefore, Sourcefire discloses this limitation.

[8.3] dropping, by the computing system, the first packet; and

231. Sourcefire discloses this limitation.

232. For the same reasons I discussed with respect to Claim 1, Sourcefire discloses that a *drop* operator causes the first packet to be dropped. See Claim [1.1], [1.3].

233. Therefore, Sourcefire discloses this limitation.

[8.4] responsive to a determination by the computing system that a second packet of the plurality of packets does not comprise data associated with XMPP:

234. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

235. For the reasons I discussed with respect to Claim 1, Sourcefire discloses that each packet would be processed by the intrusion rules and thus the IPS would make a determination if data for a second packet did not meet the XMPP criteria (*e.g.*, *non-IM* data such as HTTP, FTP, RTP, SMTP data). See Claim [1.4].

236. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[8.5] applying, by the computing system, to the second packet, and without applying the one or more packet-filtering rules configured to prevent the particular type of data transfer from the first network to the second network, a second operator configured to forward packets not associated with the particular type of data transfer toward the second network; and

237. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

238. For the reasons I discussed regarding Claim 1, Sourcefire discloses that a second packet that did not meet the XMPP criteria in the one or more packet-filtering rules enforcing the no-employee IM security policy (*e.g.*, by including *non*-IM data such as HTTP, FTP, SMTP data) would have a forward or *pass* operator applied to it. CS-1004 p. 334, 2092; See also Claim [1.4]-[1.5]. For example, for the second network, a POSA understood that an employee security policy which included the one or more packet-filtering rules blocking troublesome IM communications to an external network would also have included packet-filtering rules which permitted employees to conduct business through other (non-IM) types of data transfer communications to the external network (such as HTTP or HTTPS web-browsing, e-mail, etc.). *Id.*; See also Claim [8.1]. Moreover, while Claim 1 is directed to forwarding packets to a different “third network,” and Claim 8 forwards packets to the same “second network,” a POSA understood that this is merely a function of the destination address of the packet and that the rationale described above for Claim 1 for forwarding the packet applies equally without regard to whether the packet is destined for a second network or a third network. See also Claim [1.4]-[1.5].

239. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[8.6] forwarding, by the computing system, the second packet toward the second network.

240. Sourcefire discloses this limitation.

241. For the reasons I discussed above regarding Claims 1 and [8.5], Sourcefire discloses that a pass/forward operator causes the second packet to be forwarded. See Claims [1.6], [8.5].

242. Therefore, Sourcefire discloses this limitation.

Claim 9: [9.0] The method of claim 1, wherein: receiving the plurality of packets comprises receiving packets comprising data associated with hypertext transfer protocol (HTTP);

243. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

244. As I discussed above for Claims 1 and 4, the plurality of packets could comprise data associated with HTTP (e.g., port 80, HTTP method). See Claims [1.1], [1.4], [4.1]. As I also discussed above for Claim 1, a POSA understood Sourcefire disclosed how a user would have written a rule using the HTTP Method option of the HTTP content keyword as part of the criteria to invoke the HTTP Inspect preprocessor to identify a first portion of packets using the particular HTTP method type of data transfer (e.g., PUT, POST, CONNECT, or other known

methods such as DELETE) and to block such packets with certain application payload content posing a threat from continuing towards their second network destination. See Claim [1.1]; see also Section VII.C, ¶¶80, 82.

245. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[9.1] the first portion of packets comprises a first type of data associated with HTTP;

246. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

247. As I discussed above, a POSA reading Sourcefire understood that a first type of data associated with HTTP could be an HTTP method such as POST, PUT, DELETE or CONNECT, and thus a first portion of packets would include packets including POST, PUT, DELETE or CONNECT as the HTTP method. See Claims [1.1], [9.0].

248. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[9.2] the second portion of packets comprises a second type of data associated with HTTP; and

249. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

250. As I discussed above, a POSA reading Sourcefire understood that a second type of data associated with HTTP could be an HTTP method such as GET, and thus a second portion of packets would include packets including GET as the HTTP method. See Claim [1.1], [1.4]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44 (“[HTTP GET] method[] ought to be considered ‘safe’” compared to “other methods, such as POST, PUT and DELETE.”), 46-49, 130-134.

251. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

[9.3] applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises the first type of data.

252. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

253. As I discussed above, a POSA reading Sourcefire understood that a user could implement a specific security policy which achieved predictable benefits of protecting his or her network against a known security vulnerability by writing a rule using the HTTP Method option of the HTTP content keyword as part of the criteria to invoke the HTTP inspect preprocessor to identify traffic using that HTTP method (*e.g.*, POST, PUT, DELETE, or CONNECT) and to *drop* (*i.e.*, block) such packets with certain application payload content posing a threat (*e.g.*, sensitive data or a malicious Uniform Resource Identifier (URI)) from continuing

towards their second network destination . CS-1004 pp. 785-786, 807, 435-439, 491; Claims [1.1]; [9.0]-[9.1].

254. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 10: [10.0] The method of claim 9, wherein applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises data corresponding to an HTTP POST method.

255. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

256. For the reasons I discussed regarding Claim 9, a POSA would be motivated to use rules as taught by Sourcefire to block a first portion of packets identifying an HTTP POST method to, for example, prevent an employee from posting sensitive data to a bulletin board or providing such data to a malicious URI on the second network. See Claims [1.1], [9.0]-[9.1], [9.3]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44, 47.

257. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 11: [11.0] The method of claim 9, wherein applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises data corresponding to an HTTP PUT method.

258. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

259. For the reasons I discussed regarding Claim 9, a POSA would be motivated to use rules as taught by Sourcefire to block a first portion of packets identifying an HTTP PUT method to, for example, prevent an employee server from storing sensitive data under the specified URI on the second network. See Claims [1.1], [9.0]-[9.1], [9.3]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44, 47-48.

260. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 12: [12.0] The method of claim 9, wherein applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises data corresponding to an HTTP DELETE method.

261. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

262. For the reasons I discussed regarding Claim 9, a POSA would be motivated to use rules as taught by Sourcefire to block a first portion of packets identifying an HTTP DELETE method to, for example, prevent an employee server from deleting the data identified by a malicious URI on the second network. See Claims [1.1], [9.0]-[9.1], [9.3]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44, 48-49.

263. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 13: [13.0] The method of claim 9, wherein applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises data corresponding to an HTTP CONNECT method.

264. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

265. For the reasons I discussed regarding Claim 9, a POSA would be motivated to use rules as taught by Sourcefire to block a first portion of packets identifying an HTTP CONNECT method to, for example, prevent an employee from tunneling sensitive data to a malicious second network proxy. See Claims [1.1], [9.0]-[9.1], [9.3]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44, 49.

266. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 14: [14.0] The method of claim 9, wherein applying the first operator⁹ configured to forward packets not associated with the particular type of

⁹ I have been informed and understand that a rule of claim drafting known as “antecedent basis” states that a claim element is identified throughout a claim by designating the element with an indefinite article “a” or “an” on its first appearance in the claim and thereafter employing a definite article such as “the” or “said.”

data transfer toward the second network is performed responsive to a determination by the computing system that the second portion of packets comprises the second type of data.

267. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

268. For the reasons as described regarding Claim 9, a POSA would be motivated to use rules as taught by Sourcefire to forward a second portion of packets associated with an HTTP GET method to, for example, permit employees to surf the web via a third network web server. See Claims [1.1], [1.4]-[1.5], [9.2]; Section VII.C, ¶¶80, 82; CS-1046 pp. 44, 46.

Claim 14 refers to forwarding packets and ultimately depends from Claim 1.

Claim 1 refers to the “second operator” as the operator configured to forward packets. CS-1001, 11:55-58. Further, Claims 1, 9-13 and 16 each recite that the first operator is configured to “drop packets.” CS-1001, 11:40-42, 13:3-59. As such, for the purposes of this analysis, I have assumed that the “the first operator” and “toward the second network” in Claim 14 are typographical errors and Claim 14 reads “the second operator” and “toward the third network.” See Claims 1, 9, 14. The same errors appear in Claims 15 and 17 and I have treated these claims as if the claims recite “the second operator” and “toward the third network.” See Claims 1, 9, 14, 15, 17.

269. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 15: *[15.0] The method of claim 9, wherein applying the first operator configured to forward packets not associated with the particular type of data transfer toward the second network is performed responsive to a determination by the computing system that the second portion of packets comprises data corresponding to an HTTP GET method.*

270. Sourcefire discloses this limitation for the reasons I discussed with respect to Claim 14.

Claim 16: *[16.0] The method of claim 1, wherein applying the first operator configured to drop packets associated with the particular type of data transfer is performed responsive to a determination by the computing system that the first portion of packets comprises data corresponding to a particular transport layer security (TLS) version value.*

271. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

272. As I discussed above for Claim 1, a POSA would be motivated to use rules using the ssl_version keyword as taught by Sourcefire to block SSL/TLS Record Protocol packets that proposed using (or did use) a less secure TLS version value (e.g., TLS 1.0) from continuing towards their second network destination IP address to achieve predictable benefits of increased network security. See Claim [1.1].

273. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 17: *[17.0] The method of claim 1, wherein applying the first operator configured to forward packets not associated with the particular type of data transfer toward the second network is performed responsive to a determination by the computing system that the second portion of packets comprises data corresponding to a particular transport layer security (TLS) version value.*

274. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

275. As I discussed above for Claim 1, a POSA would be motivated to use rules using the ssl_version keyword as taught by Sourcefire to forward TLS Record Protocol packets that proposed using (or did use) a more secure TLS version value (e.g., TLS 1.1, TLS 1.2) toward their third network destination IP address to achieve predictable benefits of increased throughput and network security. See Claim [1.1], [1.4].

276. Therefore, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation.

Claim 18: *[18.0] A system comprising: at least one processor; and [18.1] a memory storing instructions that when executed by the at least one processor cause the system to: [18.2] receive, from a computing device located in a first network, a plurality of packets wherein the plurality of packets comprises a first portion of packets and a second portion of packets; [18.3] responsive to a determination that the first portion of packets comprises data corresponding to criteria specified by one or more packet-filtering rules configured to prevent a particular type of data transfer from the first network to a second network, wherein the data indicates that the first portion of packets is destined for the second network: [18.4] apply, 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; and [18.5] drop each packet in the first portion of packets; and [18.6] responsive to a determination that*

the second portion of packets comprises data that does not correspond to the criteria, wherein the data indicates that the second portion of packets is destined for a third network: [18.7] apply, to each packet in the second portion of packets, and without applying the one or more packet-filtering rules configured to prevent the particular type of data transfer from the first network to the second network, a second operator configured to forward packets not associated with the particular type of data transfer toward the third network; and [18.8] forward each packet in the second portion of packets toward the third network.

277. Claim 18 is a system claim corresponding to the method recited in Claim 1 and therefore each of the limitations recited in Claim 18 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the reasons I discussed with respect to Claim 1. See Claim [1.0]-[1.6]. Sourcefire describes that each 3D Sensor included a processor (CPU), memory, and disk storage. CS-1004 pp. 33-34, 106-107.

Claim 19: [19.0] 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: [19.1] receive, from a computing device located in a first network, a plurality of packets wherein the plurality of packets comprises a first portion of packets and a second portion of packets; [19.2] responsive to a determination that the first portion of packets comprises data corresponding to criteria specified by one or more packet-filtering rules configured to prevent a particular type of data transfer from the first network to a second network, wherein the data indicates that the first portion of packets is destined for the second network: [19.3] apply, 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; and drop each packet in the first portion of packets; and [19.4] responsive to a determination that the second portion of packets comprises data that does not correspond to the criteria, wherein the data indicates that the second portion of packets is destined for a third network: [19.5] apply, to each packet in the second

portion of packets, and without applying the one or more packet-filtering rules configured to prevent the particular type of data transfer from the first network to the second network, a second operator, configured to forward packets not associated with the particular type of data transfer toward the third network; and [19.6] forward each packet in the second portion of packets toward the third network.

278. Claim 19 is a computer program claim corresponding to the method recited in Claim 1 and therefore each of the limitations recited in Claim 19 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the reasons I discussed with respect to Claim 1. See Claim [1.0]-[1.6].

Sourcefire describes that each 3D Sensor included a processor (CPU), memory, and disk storage. CS-1004 pp. 33-34, 106-107. A POSA understood that the 3D Sensor disk memory is non-transitory computer-readable media that included instructions to control the operation of the 3D Sensor. CS-1004 pp. 33-34, 106-107.

Claim 20: [20.0] The method of claim 1, wherein: the first portion of packets comprises data indicating: a first source port number associated with the particular type of data transfer, and corresponding to the criteria specified by the one or more packet-filtering rules; and [20.1] the second portion of packets comprises data indicating a second source port number not associated with the particular type of data transfer.

279. Sourcefire in view of the knowledge, skill, and creativity of a POSA discloses this limitation.

280. Similar to the reasons I discussed regarding Claims 1 and 3, the first portion of packets could comprise data (*e.g.*, 5-tuple source port (80), etc.) indicating an HTTP protocol type and that a web server on a first network (*e.g.*, external network) seeks a response from a client on a second network (*e.g.*, internal network) associated with a particular type of data transfer (*e.g.*, HTTP PUT) and corresponding to the criteria specified by the one or more packet-filtering rules (*e.g.*, 5-tuple and keyword rule criteria). See Claims [1.1], [3.1]. Likewise, the second portion of packets could comprise data indicating a second source port number (*e.g.*, port 25) *not* associated with the particular type of data transfer (*e.g.*, SMTP) and a third network destination IP address. See Claims [1.4]-[1.5], [3.2]; CS-1004 p. 639.

281. Therefore, Sourcefire in view of the knowledge, skill, and creativity of a POSA discloses this limitation.

XI. Conclusion and Declaration

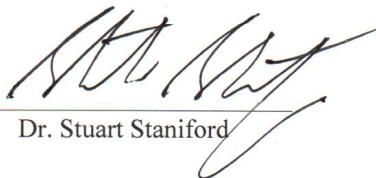
282. The findings and opinions set forth in this declaration are based on my work and examinations to date. I may continue my examinations. I may also receive additional documentation and other factual evidence over the course of this litigation that will allow me to supplement and/or refine my opinions. 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.

283. 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.

284. 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: August 15, 2018

By: 
Dr. Stuart Staniford

Stuart Staniford

317 N. Aurora St, Box #1,
Ithaca, NY 14850,
stuart@staniford.systems

Summary

I am an inventor and entrepreneur with two decades of experience innovating in network security startups. I was an executive and a key inventor during the rise of FireEye, which was crowned the “*Hottest Security Startup in Silicon Valley*” by Forbes Magazine, went IPO in September 2013, and subsequently reached a market value of \$10b. I have innovated in detecting web application attacks, Chinese APT attacks, web drive-by attacks, and network worm propagation. I have raised millions in funding, spent years on executive management teams, sold into Fortune 500 accounts, closed seven figure deals, and interacted with boards. My early security research papers have been cited over 7000 times in the scientific literature, as well as being covered in major national magazines and newspapers. I have been an expert witness and testified at trial. I have extensive hands-on development experience in C, Javascript, Java, C++, Python SQL, and Perl, and am intimate with HTML/JS and HTTP.

Experience

Staniford Systems. President, April 2017 – Present

Independent technology consultant for various clients.

Viridius Property. President, April 2014 – Present

Developed this real estate company providing fossil-fuel free rentals for both residential and commercial markets.

Cayuga Networks. CEO/CTO, April 2014 – Feb 2017.

Founded this startup in Ithaca, NY, developing a high-speed network web application attack detection system. Recruited engineering, office, and executive staff, supervised setup of initial systems, instituted continuous integration development methodology and raised \$2.3m in outside venture capital. Sold into Fortune 500 accounts. Managed a team of 20. Company failed to establish sufficient traction in time.

FireEye. Chief Scientist, Jan 2008 – Feb 2013.

Developed 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 problem. Company successfully raised two rounds of venture capital based on this new value proposition, sold into numerous Fortune 500 accounts, and began doubling and tripling in size annually, eventually reaching over \$10b in valuation and being named the “*Hottest Security Startup in Silicon Valley*” by Forbes magazine. I also invented and developed a PDF parser and zero-day detection module to address shift of web attacks to primarily

PDF vector, and then did the same for Java and SWF. Performed analysis of our global data for marketing and scientific purposes. Discovered significant deployment issues at a large minority of our customers and led resolution effort. FireEye's success obligated top analysts Gartner to create an entire new market category, Advanced Threat Protection Appliances, which recognized FireEye as the "first company to bring to market automation of attack detection and prevention using virtual execution of objects and analysis." FireEye successfully went public in September 2013, and the share price went from 7c when I joined to a peak over \$95 as of early 2014. 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.

Cornell University. Adjunct Professor of Computer Science, Jul 2013 – Jun 2016.

Taught graduate course on Defending Computer Networks for three years.

Invicta Consulting. President, 2005 - 2007

Solo consultant. Technical consulting for several startups – found security vulnerabilities and false negatives in products, designed algorithms, analyzed competing products for effectiveness. Also worked as litigation consultant and expert witness for several law firms – analyzed patents and prior art, wrote expert report, defended analysis at deposition, testified at trial.

Nevis Networks. Principal Scientist, April 2004 – July 2005

Architected a very high-speed event correlation system and the traffic anomaly subsystem for this startup developing 10Gbps network security solutions for the ethernet edge of internal enterprise networks. These systems resulted in four patent applications including a novel multi-dimensional external memory algorithm for storing log-records on disk at very high speeds. Designed portions of the product's graphical user interface. Worked extensively with engineering teams in Pune, India implementing my designs.

Silicon Defense. Founder and President, 1998 - 2004

Managed 23 staff performing a mixture of government contract research and commercial product development. Obtained ten research contracts for the company up to \$2.3m in size, working for four different DARPA program managers. Wrote business plan for the company and raised \$300k in angel capital. Sold commercial products into Fortune 500 accounts (company gained over 50 commercial customers during my tenure, including Disney, AT&T, Hartford Insurance, and Invesco). Had profit and loss responsibility for a \$2.3m operation, and extensive experience interacting with press, analyst, and investment communities. Also led two standards groups, and served on a number of program committees. After five years of 100%+ annual growth out of cashflow, company was obliged to file bankruptcy due to DARPA's decision to classify further information security research.

UC Davis. Researcher, 1994 – 1997, and Assistant Adjunct Professor, 1997 - 1999

Founded and co-chaired the working group that developed the Common Intrusion Detection Framework at the request of DARPA. This involved working with a team of

over a hundred researchers and developers from a wide variety of companies and organizations. Led a team of ten researchers and students building a large, distributed, intrusion-detection system (GrIDS). Performed research in new statistical techniques to help in tracing intruders across the Internet. Presented work at conferences and to funding agencies. Wrote successful funding proposals and published papers on work.

Education

M.S. (Computer Science). March 1995. UC Davis. Advisor: Prof. Karl Levitt

Ph.D. (Physics) June 1993. UC Davis.

M.S. (Physics) June 1990. UC Davis.

B.Sc. (Mathematical Physics) June 1988. University of Sussex, UK. First Class Honors.

Refereed Publications

S. Staniford, D. Moore, N. Weaver, and V. Paxson, *The Top Speed of Flash Worms*. Proceeding of the ACM Workshop on Rapid Malcode (WORM), 2004

N. Weaver, D. Ellis, S. Staniford, and V. Paxson, *Worms vs Perimeters – The Case for Hard-LANS*. Proceedings of Hot Interconnects, 2004

N. Weaver, V. Paxson, and S. Staniford, *Very Fast Scanning Worm Containment*. Proceedings of USENIX Security, 2004

S. Staniford. *Containment of Scanning Worms in Enterprise Networks*. To appear in the Journal of Computer Security.

N. Weaver, V. Paxson, S. Staniford, and R. Cunningham *A Taxonomy of Computer Worms*. Proceedings of the ACM Workshop on Rapid Malcode (WORM). Washington D.C. October, 2003.

D. Moore, V. Paxson, S. Savage, C. Shannon, S. Staniford, and N. Weaver, *Inside the Slammer Worm*, IEEE Security and Privacy, July/August 2003.

S. Staniford, J. Hoagland and J. McAlerney. *Practical Automated Detection of Stealthy Portscans*. Journal of Computer Security. Vol 10, Issue 1/2, 2002.

S. Staniford, V. Paxson, and N. Weaver *How to Own the Internet in Your Spare Time*, Proceedings of the 11th USENIX Security Symposium 2002.

D. Donoho, A. Flesia, U. Shankar, V. Paxson, J. Coit, and S. Staniford, *Multiscale Stepping-Stone Detection: Detecting Pairs of Jittered Interactive Streams by Exploiting Maximum Tolerable Delay*, Proc. RAID 2002.

J. Hoagland, and S. Staniford, *Viewing IDS alerts: Lessons from SnortSnarf*. Proceedings of DISCEX II, Anaheim, June 2001.

J. Coit, S. Staniford, and J. McAlerney. *Towards Faster Pattern Matching for Intrusion Detection: Exceeding the Speed of Snort*. Proceedings of DISCEX II, Anaheim, June 2001.

R. Feiertag, S. Rho, L. Benzinger, S. Wu, T. Redmond, C. Zhang, K. Levitt, D. Peticolas, M. Heckman, S. Staniford-Chen, and J. McAlerney, *Intrusion Detection Inter-component Adaptive Negotiation*. Computer Networks. 2000.

S. Staniford, J. Hoagland and J. McAlerney. *Practical Automated Detection of Stealthy Portscans.* Proceedings of the ACM CCS IDS Workshop, November 1, 2000. Athens, Greece.

R. Feiertag, S. Rho, L. Benzinger, S. Wu, T. Redmond, C. Zhang, K. Levitt, D. Peticolas, M. Heckman, S. Staniford-Chen, and J. McAlerney, et al. *Intrusion Detection Inter-Component Adaptive Negotiation.* Proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID 99), Lafayette, Indiana; September, 1999.

S. Staniford-Chen, B. Tung, and D. Schnackenberg. *The Common Intrusion Detection Framework (CIDF).* Proceedings of 1998 Information Survivability Workshop – ISW’98, Orland, Florida; October, 1998.

S. Staniford-Chen, S. et al [*GrIDS: A Graph-Based Intrusion Detection System for Large Networks.*](#) Proceedings of the 19th NISSC, Baltimore, 1996.

S. Staniford-Chen, and L.T. Heberlein, [*Holding Intruders Accountable on the Internet.*](#) Proceedings of the 1995 IEEE Symposium on Security and Privacy, Oakland, CA. 1995.

J. Kiskis and S. Staniford-Chen, *Universal Amplitude Ratios and Functions for the SU(2), Finite-Temperature Phase Transition.* In Axen, D., Bryman, D., and Comyn, N. (eds) Vancouver Meeting. Particles and Fields '91. p 821. World Scientific. 1992.

Published Reports and Theses

N. Weaver, V. Paxson, and S. Staniford, *The Worst Case Worm.* Silicon Defense Technical Report. August 2003.

S. Staniford and C. Kahn, *Worm Containment on the Internal Network.* Silicon Defense Technical White Paper. March 2003

N. Weaver, V. Paxson, S. Staniford, and R. Cunningham, *Large Scale Malicious Code: A Research Agenda.* Silicon Defense Technical Report, Dec 2002.

B. Tung, et al. The Common Intrusion Detection Framework Specification. Nov 2001.

S. Staniford, O.S. Saydjari, and K. Williams *The US is Not Safe in a Cyberwar.* Paper presented to Department of Defense and National Security Council executives. May 2001. 2nd Edition.

S. Staniford, O.S. Saydjari, and K. Williams *The US is Not Safe in a Cyberwar.* Paper presented to DARPA. Sep 2000.

S. Cheung, S. et al [*The Design of GrIDS: A Graph-Based Intrusion Detection System.*](#) UCD Technical Report CSE-99-2, January, 1999.

S. Staniford-Chen, [*Distributed Tracing of Intruders.*](#) Master's Thesis, University of California at Davis. 1995.

S. Staniford-Chen, Finite Size Scaling and the Universality Class of SU(2) Lattice Gauge Theory. PhD Thesis, University of California at Davis. 1993.

S. Staniford-Chen, Finite Size Scaling of Probability Distributions in SU(2) Lattice Gauge Theory and Φ^4 Field Theory . Preprint UCD-92-17, University of California at Davis. 1992.

Patents and Patent Filings

S. Staniford and A. Aziz, *Systems and methods for analyzing PDF documents*. Patent #9,954,890. April 24th, 2018.

A. Aziz, S. Staniford, M. Amin, H. Uyeno, and S. Yie, *Detecting malicious network content*. Patent #9,519,782. December 13th, 2016.

S. Staniford and A. Aziz, *Systems and methods for analyzing malicious PDF network content*. Patent #9,438,622. September 6th, 2016.

S. Staniford and A. Aziz, *Systems and methods for detecting malicious PDF network content*. Patent #9,118,715. August 25th, 2015.

A. Aziz, H. Uyeno, J. Manni, M. Amin, and S. Staniford, *Electronic message analysis for malware detection*. Patent #9,106,694. August 11th, 2015.

S. Staniford and A. Aziz, *Systems and methods for detecting malicious PDF network content*. Patent #8,997,219. March 31st, 2015.

S. Staniford and A. Aziz, *Systems and methods for scheduling analysis of network content for malware*. Patent #8,990,939. March 24th, 2015.

S. Staniford and A. Aziz, *Systems and methods for detecting malicious network content*. Patent #8,850,571. September 30th, 2014

S. Staniford and M. Bakshi, *A System and Method for Selecting Memory Locations for Overwrite*. Filed January 23rd, 2006

S. Staniford et al, *A System and Method for Aggregating and Consolidating Security Event Data*. Filed November 26th, 2005

S. Staniford and T. Mustafa, *A System and Method for Deprioritizing and Presenting Data*. Filed November 4th, 2005

S. Staniford and P. Sobel, *System and method for storing multi-dimensional network and security event data*. Filed October 14th, 2005

S. Staniford, C. Kahn, N. Weaver, C. Coit, and R. Jonkman, *Method and system for reducing the rate of infection of a communications network by a software worm*. Filed December 6th, 2002. Filing serial number: 313623.

Litigation Consultant/Expert Witness Matters

I have been asked to provide expert opinion in the following litigation matters:

Nomadix vs Guest-Tek. 2018. Law Firm Knobbe Martens for Nomadix. Consulted on litigation positions.

Deep Nines vs McAfee. 2007. Law Firm Williams Morgan for McAfee. Researched prior art, prepared claim charts.

SRI vs ISS. 2005-2008. Law Firm King and Spalding for ISS. Research prior art, argued patent invalidity and non-infringement on one of the patents in suit. Wrote expert report, was deposed, testified at trial. The client won on the patent I worked on.

Selected Press Coverage of Work

My work has been featured in several dozen stories in major media and technical publications. A small sample include:

New York Times Thieves Winning Online War, Maybe Even in Your Computer

<http://www.nytimes.com/2008/12/06/technology/internet/06security.html>

A research report last month by Stuart Staniford, chief scientist of FireEye, a Silicon Valley computer security firm, indicated that in tests of 36 commercial antivirus products, fewer than half of the newest malicious software programs were identified.

ComputerWorld Antivirus no defense against botnets, says vendor

<http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9121901>

A new analysis of botnets has come up with a possible reason for their prodigious ability to infect PCs: Many antivirus programs are near to useless in blocking the binaries used to spread them.

Atlantic Magazine Running Dry

<http://www.theatlantic.com/doc/200710/oil-field-decline>

The world's most essential oil field may be in decline.

Business Week To Trap a Superworm

http://www.businessweek.com/technology/content/feb2003/tc20030225_4104_tc047.htm

The Slammer worm's ability to spread so rapidly adds a frightfully new dimension to the species. Does Stuart Staniford have the cure?

PC World Dawn of the Superworm,

<http://www.pcworld.com/news/article/0,aid,110014,00.asp>

ComputerWorld Study: Slammer was fastest spreading worm yet,

<http://www.idg.com.hk/cw/readstory.asp?aid=20030205005>

The Independent Internet worm took 10 minutes to create global chaos,

<http://news.independent.co.uk/digital/news/story.jsp?story=375374>