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. 9,124,552 to Sean Moore
Issue Date: September 1, 2015
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. 9,124,552

***Mail Stop* "*PATENT BOARD*"**
Patent Trial and Appeal Board
U.S. Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

TABLE OF CONTENTS

Page

I. Introduction 1

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

III. My Background and Qualifications 8

IV. Person of Ordinary Skill in the Art (POSA) 12

V. Relevant Legal Standards 13

VI. The ’552 Patent 17

A. Description 17

B. Priority Date of Claims 27

C. Prosecution History of the ’552 Patent 27

VII. State of the Art 29

A. Packet-Switched Networking and Communication Protocols 30

B. Network Security Systems - Firewalls 32

C. Network Security Systems –Intrusion Detection Systems and Intrusion Prevention Systems 36

D. Inspecting and Applying Packet Filtering Rules to Encrypted and Unencrypted SSL/TLS Record Protocol Data 39

VIII. Claim Construction 50

IX. Overview 51

X. Challenge: Claims 1-21 are Obvious under 35 U.S.C. § 103 over Sourcefire in View of the Knowledge and Skill of a POSA 52

A. Sourcefire Teaches Every Element of Claims 1-21 in View of the Knowledge and Skill of a POSA 52

1. Sourcefire 52

B. Detailed Invalidity Analysis.....65

XI. Conclusion and Declaration.....103

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. 9,124,552 ("the '552 Patent") (CS-1001). The '552 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 '552 Patent issued September 1, 2015, from U.S. Patent Application No. 13/795,822 which was filed March 12, 2013. 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 '552 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 ’552 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. CS-1001, 1:15-37. I am familiar with the technology described in the ’552 Patent as of the earliest possible priority date of March 12, 2013.

6. In preparing this Declaration, I have reviewed the ’552 Patent (CS-1001), the file history of the ’552 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 ’552 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 ’552 Patent. As described in detail below, I offer the following opinion in this Declaration:

- A POSA would have found the invention of Claims 1-21 of the ’552 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. 9,124,552 (“the ’552 Patent”)
CS-1002	U.S. Patent No. 9,124,552 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. Pat. No. 8,370,936 (“Zuk”)
CS-1009	U.S. Pat. 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 26, 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	An Overview of SSL (version 2). Adam Shostack, 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," *IEEE Communications Magazine*, September 1994
CS-1020	Barry M. Leiner *et al*., "A Brief History of the Internet". *Internet Society*, 1997, available at: https://www.internetsociety.org/internet/history-internet/brief-history-internet/ (accessed June 18, 2018).
CS-1021	"Internet Protocol," *IETF RFC 791*, J. Postel, ed., September 1981, archived at Internet Archive Wayback Machine on February 4, 2007 at https://web.archive.org/web/20070204151303/https://tools.ietf.org/html/rfc791.
CS-1022	"Transmission Control Protocol," *IETF RFC 793*. J. Postel, ed. September 1981, archived at Internet Archive Wayback Machine on February 2, 2007 at https://web.archive.org/web/20070202201546/https://tools.ietf.org/html/rfc793.
CS-1023	E. H. Spafford, "The Internet Worm Program: An Analysis," *Purdue Technical Report CSD-TR-823*, 1988.

Exhibit No.	Title
CS-1024	Eichin, M.W. and Rochlis, J.A., "With Microscope and Tweezers: An Analysis of the Internet Virus of November 1988," *Proceedings, 1989 IEEE Symposium on Security and Privacy*, (pp. 326-343), IEEE, 1989.
CS-1025	T. Eisenberg *et al.*, "The Cornell Commission: On Morris and the Worm," *Communications of the ACM*, Volume 32 Number 6, June 1989.
CS-1026	J. C. Mogul, "Simple and Flexible Datagram Access Controls for Unix-based Gateways," *Digital Western Research Laboratory Technical Report No. 89/4*, 1989.
CS-1027	Heberlein, L. T.L, *et al.*, "A Network Security Monitor," *Proceedings, 1990 IEEE Computer Society Symposium on Security and Privacy,* 1990.
CS-1028	"NetRanger Network Security Management System User Guide". v1.3.1 *Wheelgroup Corporation*, 1997.
CS-1029	Roesch, M., "Snort: Lightweight Intrusion Detection for Networks," *Lisa* (Vol. 99, No. 1, pp. 229-238), 1999.
CS-1030	Brandon De Hoyos, "IRC, ICQ, AIM and More: A History of Instant Messaging," *Lifewire*, 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," *IETF RFC 1919*, March 1996.
CS-1032	K. Ingham & S. Forrest, "A History and Survey of Network Firewalls," *University of New Mexico Computer Science Department Technical Report 2002-37*, 2002.
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").

Exhibit No.	Title
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," *IETF RFC 765*, J. Postel, ed., June 1980, archived at Internet Archive Wayback Machine on February 6, 1997 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," *World Wide Web Consortium*, 1991, archived at Internet Archive Wayback Machine on June 5, 1997 at https://web.archive.org/web/19970605071155/https://www.w3.org/Protocols/HTTP/AsImplemented.html.
CS-1037	I. Ristic, "SSL/TLS and PKI History," April 2018. Available from https://www.feistyduck.com/ssl-tls-and-pki-history/, (accessed June 27th, 2018).
CS-1038	E. Rescorla, "HTTP over TLS", *IETF RFC 2818*, May 2000, archived at Internet Archive Wayback Machine on January 25, 2007 at https://web.archive.org/web/20070125003236/https://tools.ietf.org/html/rfc2818.
CS-1039	J. Moser, "The First Few Milliseconds of an HTTPS Connection", *Moserware*, June 10, 2009, archived at Internet Archive Wayback Machine on June 13, 2009 at https://web.archive.org/web/20090613041022/http://www.moserware.com/2009/06/first-few-milliseconds-of-https.html

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 Sections IV, 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 '552 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, communication protocols, 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 '552 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 ’552 Patent

A. Description

33. The ’552 Patent is directed to systems and methods for filtering data packets leaving a network to ensure that the data packets are not improperly transferring sensitive data or credentials from the network, a category of cyber-attacks known as exfiltrations. For example, exfiltrations are often facilitated by using popular network data transfer protocols, such as the Hypertext Transfer Protocol (HTTP) used by the World Wide Web. Such attacks may appear to an insufficiently careful observer as normal network behavior; older network trust models, such as those used by firewalls that do not interpret application layer data like HTTP, may interpret exfiltrations as trusted operations. CS-1001 1:15-37.

34. The ’552 Patent describes a two-stage filtering process for stopping such attacks implemented at a Packet Security Gateway (PSG) examining packets that are exiting a trusted network towards an external network. CS-1001 8:19-31.

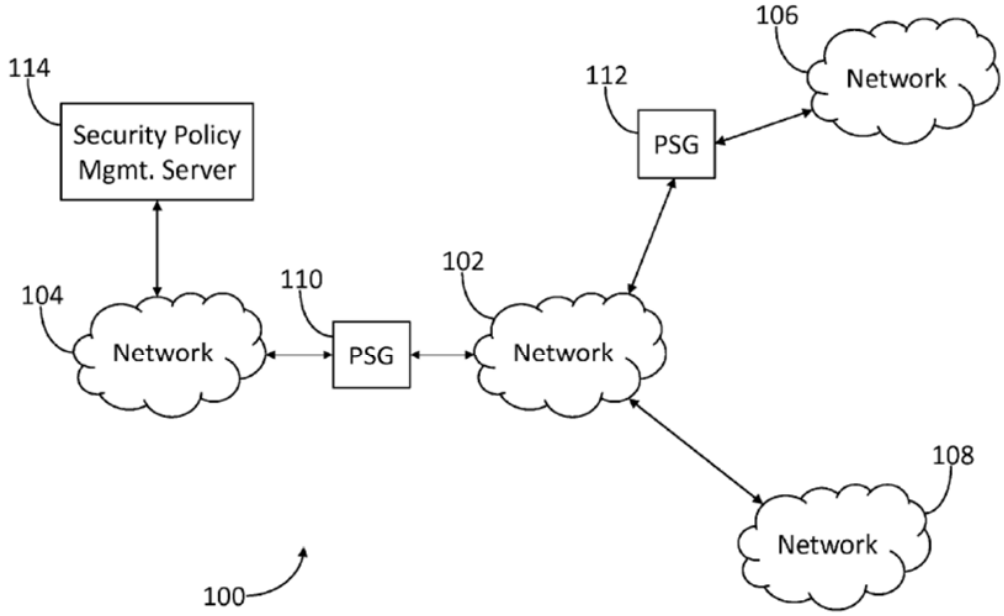


FIG. 1

35. FIG. 1 of the ’552 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* 3:12-26. 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 3:61-4:5; 3:42-60; 5:6-27.

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 4:6-19.

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 8:19-30.

38. FIG. 4 of the '552 Patent (below) illustrates the two-stage filtering process performed at a PSG:

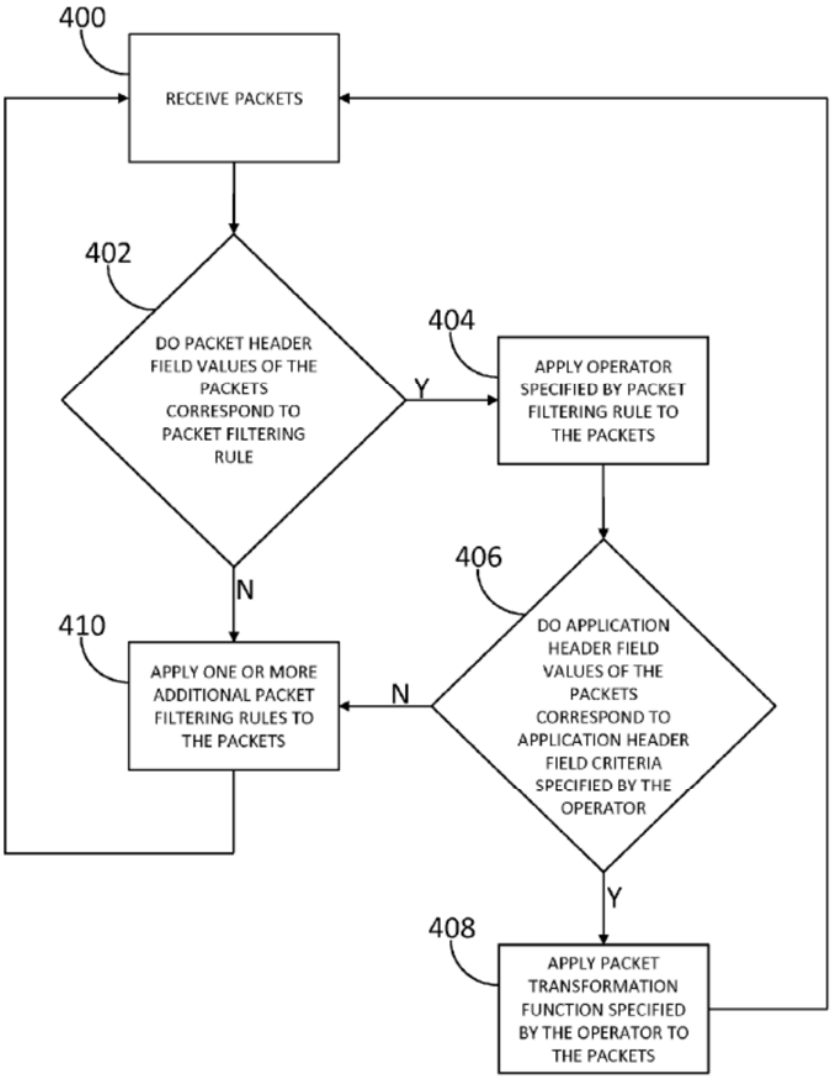


FIG. 4

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 8:63-9:8.

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 9:9-12. FIG. 3 of the '552 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 9:16-20.

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 '552 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 7:20-24; 9:26-40; Claims 1, 2.

43. In one embodiment, the '552 Patent describes that an operator can include (1) determine if an IP packet traversing a boundary contains an HTTP packet (*e.g.*, an application-level HTTP packet) that specifies one or more specific

HTTP methods (*e.g.*, GET, PUT, POST, etc.), and (2) allow the packet *(e.g.*, if a GET method is specified), or block the packet (*e.g.*, if a PUT or POST method is specified). CS-1001 2:21-27; FIG. 3 (below) (rule 6).

44. FIG. 3 of the '552 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 5:28-42.

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* 5:42-67.

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 6:1-9. 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 6:37-47, 7:18-23, 7:54-8:18. In the specific embodiment disclosed for Rule 5, the '552 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 7:55-60.

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 ’552 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 6:48-67.

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 6:10-16.

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 7:17-23, 9:26-40.

51. By enforcing sets of rules similar to rules 1-7, the ’552 Patent states that companies may allow their employees to freely surf the web and conduct

company business, but may block one or more cyber-attacks (*e.g.*, HTTP-mediated exfiltrations). CS-1001 7:12-16.

52. The claims of the ’552 Patent are directed to a method, apparatus and computer program for implementing the two-stage packet filtering process at the PSG. Independent Claim 1 is directed to the method; independent Claim 8 is a corresponding apparatus claim performing the steps recited in Claim 1; and independent Claim 15 is a corresponding computer program programmed to perform the steps recited in Claim 1. Therefore, Claim 1 is representative of the independent claims.

53. Each of the two-stage packet filtering embodiments of the ’552 Patent discloses the following combination to enable increased network security: (1) determining whether 5-tuple packet header field values of received packets correspond to packet header criteria of a rule, (2) determining whether application header field values of the received packets correspond to application header criteria of the rule, and (3) allowing or blocking the received packets based on these determinations and the operator specified in rule. CS-1001 2:21-27, 6:37-47, 6:48-67, 7:35-8:21; FIG. 3 (rules 5, 6). The ’552 Patent describes that this same combination was used to allow or block received packets using rules with various types of application header criteria including: (a) particular HTTP methods (CS-1001 2:21-27, 6:48-67, 7:35-57, FIG. 3 (rule 6)), (b) particular encryption protocol

(TLS) versions (CS-1001 6:37-47, 7:58-8:21, FIG. 3 (rule 5)), and other application header criteria (*e.g.*, particular data transfer protocols (*e.g.*, FTP, RTP), particular messaging protocols (*e.g.*, XMPP, etc.) (*Id.*, 8:32-55).

B. Priority Date of Claims

54. As I discuss in Section VI.C, the '552 Patent application (CS-1002) was filed on March 12, 2013 and does not claim priority to any other application. CS-1002 at pp. 2-39; CS-1001 at p. 1.

55. 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 '552 Patent

56. The independent claims as originally filed were directed to the two-stage packet filter that I discussed above and which is the focus of the description of the '552 Patent. The independent claims broadly claimed uses of application header values in filtering packets, when combined with also having examined the packet header data. The dependent claims were directed to the specific embodiments described by Rules 5 and 6 from Figure 3. The independent claims were rejected as anticipated by U.S. Patent Application No. 2007/0083924 which disclosed two-stage packet filtering based on packet field header values and application header values. CS-1002 pp. 212-221. The dependent claims directed

to the specific embodiment of Rule 5 were allowed. CS-1002 p. 221. In response, the applicant amended the independent claims to include the limitation recited in the allowed dependent claims directed to Rule 5. CS-1002 pp. 254-269. A notice of allowance issued in response to the applicant's amendments. CS-1002 pp. 275-279. Thus, the applicant narrowed the original independent claims to claim subject matter disclosed in a single line of one figure in the specification (CS-1001 Fig. 3 Rule 5), and the small accompanying part of the disclosure (CS-1001 7:49-66, 8:1-18).

57. Although the USPTO examiner found patentability in rules specifying "TLS version value" criteria over rules specifying other application header criteria (See Section VI.C below), the '552 Patent does not make any distinction that the second-stage packet-filtering use of TLS version value rule criteria is more technologically challenging than use of other application header rule criteria such as other encryption protocol (*e.g.*, SSL) versions.

58. Rather, the '552 Patent treats all second-stage rule criteria the same and identifies *the exact same system architecture and method* for allowing or blocking received packets using each of the different types of application header rule criteria. Indeed, the '552 Patent describes that the exact same system architecture and method is used to allow or block packets with application header field values specifying certain data *transfer* protocols (*e.g.*, HTTP) is also used to

allow or block packets with application header field values specifying data *messaging* protocols (*e.g.*, XMPP) and data *encryption* protocols (*e.g.*, TLS 1.0).

VII. State of the Art

59. 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 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 '552 Patent. This section provides additional motivation to combine the teachings of the Sourcefire reference (CS-1004) with the knowledge, skill, and creativity of a POSA, provides support for why a POSA would have had a reasonable expectation of success in doing so, rebuts any assertion by the Patent Owner of unpredictability in the art, and rebuts any assertion by the Patent Owner of unexpected results.

60. 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.

A. Packet-Switched Networking and Communication Protocols

61. 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.

62. 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 cut over to using the new TCP/IP protocol suite on January 1, 1983. That date can be viewed as the birth date of the modern Internet. CS-1020.

63. Logically, the Internet is a web of intermediate network nodes and communication paths interconnecting those network nodes that provide multiple data transmission routes from any given point to any other given point on the network (*i.e.*, between any two computers.) 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.

64. While TCP/IP provided the basic abstract facility to move data across a global network, higher-level application protocols were required to allow data communication to be managed conveniently. Application protocol data would be transmitted in the body of packets with IP and TCP headers. Particularly important applications included email (with protocols such as SMTP, POP and IMAP), with early email systems dating back to 1972. CS-1020. Other important protocols included FTP for making files available, first defined in 1980 (CS-1035), and the Hypertext Transfer Protocol (HTTP) for facilitating the exchange of hypertext markup language (HTML) and World Wide Web (WWW) content, first defined in 1991 (CS-1036). Similarly, Telnet and then SSH were developed for remote computer terminal access, and a variety of instant messaging protocols became available for text-based communication (*e.g.*, AOL Instant Messenger (AIM) and ICQ, Windows Live Messenger, Yahoo! Messenger, Jabber using XMPP protocol, etc.). CS-1030.

B. Network Security Systems - Firewalls

65. 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.

66. 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.

67. 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.

68. 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.

69. 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 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.

70. Early packet filters reasoned packet-by-packet without keeping any state information about prior packets. It quickly became apparent that this strategy was both expensive (in that possibly large firewall rulesets had to be separately processed for every packet, even in a long connection) and had security weaknesses (as hackers found tricky ways to fool packet filters into allowing probe packets through because the filter didn't know the state of the TCP connection). This led to the development of stateful inspection firewalls. Early stateful inspection firewalls applied the rules to the first packet in a TCP connection (the "syn packet"), and then kept track in an in-memory hash table of allowed connections and their current state, meaning that subsequent packets in the connection could be allowed or blocked based on a single hash lookup. CS-1032; CS-1008 4:12-30.

71. By the mid-1990s, multiple systems had been developed which incorporated application functionality "transparently" into stateful inspection firewalls. CS-1031 (published in March 1996). The way this worked is that the firewall would parse the application data inside packets it was passing, and investigate whether they might be significant to application traffic routing. For example, the Zuk patent, filed in 2002, stated that:

> A packet filtering firewall with stateful inspection also has the ability to examine a packet in order to allow certain types of commands within an application while disallowing others. For example, a stateful inspection firewall can allow the FTP "get" command while disallowing the "put" command." CS-1008 4:31-41.

72. Based on understanding the application protocol, the firewall determined that certain TCP connections needed to be routed through a proxy. It would instantiate the relevant proxy code in its own memory and then connect the incoming client TCP connection to that proxy automatically, which would then perform authentication and provide proxy service on to the server. Thus, no later than 1996, a POSA familiar with then current firewall technology would know that some firewalls were discriminating whether to pass or block packets based on examining both the 5-tuple (*e.g.*, TCP/IP headers), but also the application headers in the packet data bodies.

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

73. 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.

74. 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. 4-80.

75. 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. 4-80.

76. 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!";)
```

77. 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)

matches a uniform resource locator (URL) used in a popular attack of the time, but content keywords in general could be used to find and match any text in application data in packets transiting the network.

78. 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.

79. 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, meant that the industry quickly developed systems that were "inline" in the traffic flow, and which could choose to drop or pass any given packet. These systems were known as intrusion prevention systems (IPS), with multiple companies launching such products in 2002, including Intruvert (later acquired by McAfee), Tipping Point (3Com then HP), and OneSecure (later acquired by NetScreen then Juniper).

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

81. The early versions of HTTP transmitted their data unencrypted ("in the clear"), leaving them vulnerable to eavesdropping and the like. To address this vulnerability, in 1995, Netscape Corporation released an implementation of the Secure Sockets Layer (SSL) protocol, which inserted an encrypted channel running on top of TCP, and underneath HTTP. The first public version of this SSL protocol, version 2, had security issues[1], 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 finially [*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 protect against CBC attacks. . . ." CS-1015 pp. 4-5; See also *id.*, p. 20.

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

[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.

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

83. 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 pp. 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)

84. 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.

85. As illustrated in the below figure, the SSL/TLS Record header added to every SSL/TLS "Record" (*i.e.* the block formed using the above-described SSL/TLS Record protocol) includes the following three fields: (1) Byte 0:

SSL/TLS Record type, (2) Bytes 1-2: SSL/TLS version (major/minor)[2], and (3) Bytes 3-4: Length of data in the Record (excluding the header itself):

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

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

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

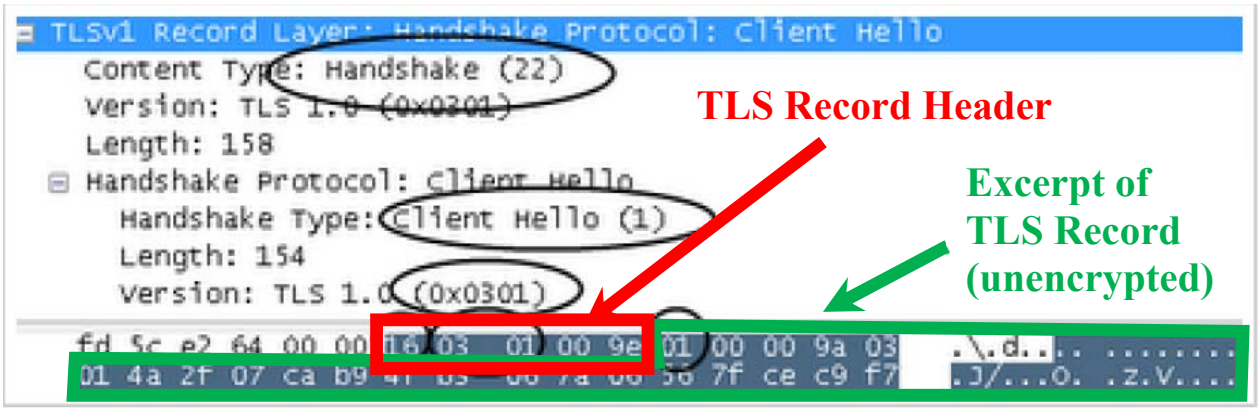
CS-1015, 16-17.

86. 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

---

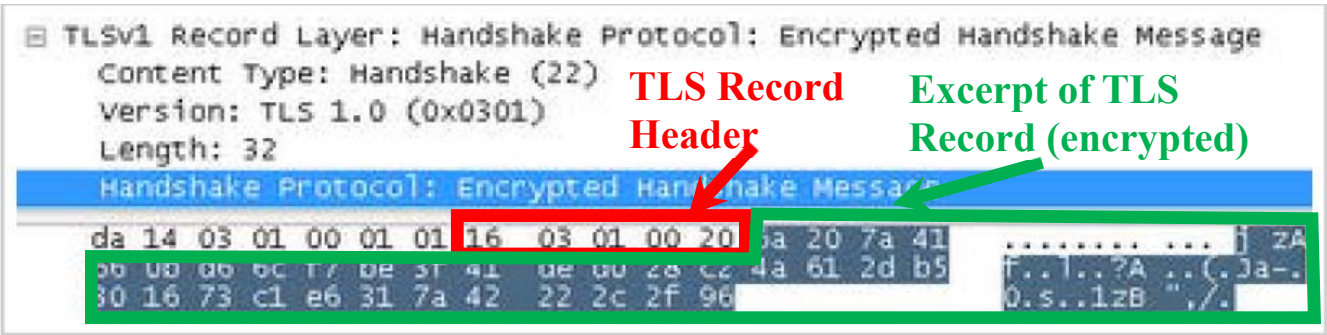
[2] In SSL v2 Record Protocol, the SSL Record header has no SSL version (CS-1018) and a POSA familiar with the relevant art would thus know that the version could only be obtained from looking at the unencrypted body (*i.e.* unencrypted SSL Record) of the SSL v2 handshake message packets. CS-1018.

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



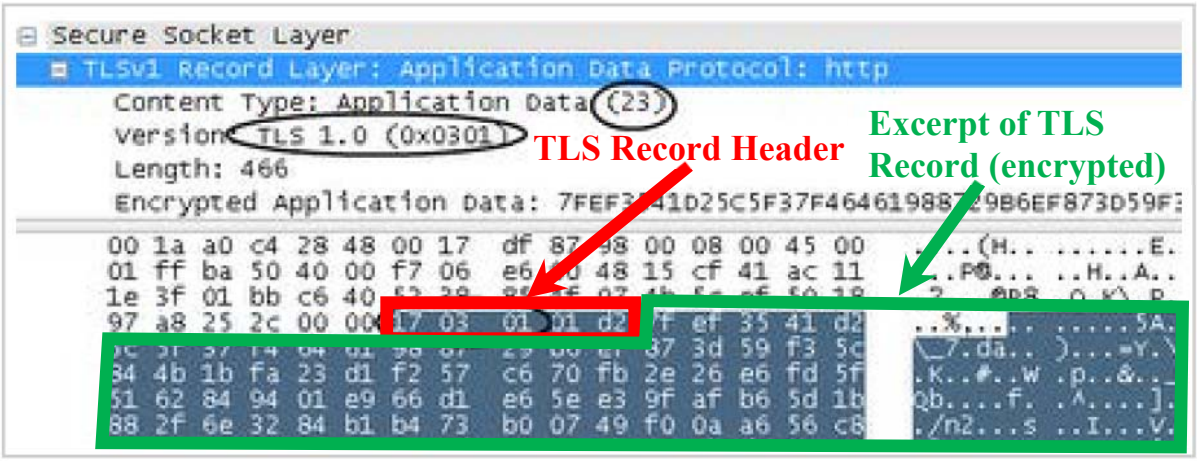
CS-1038, 3; CS-1039, 1-2. As shown in the above example, the TLS Record header specifies that the TLS Record type is Handshake (0x16 or 22), that the TLS version is TLS 1.0 (03x01 or 3,1), and that the length of unencrypted data in the TLS Record is 154 bytes (00x9e).

87. 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, 3; CS-1039, 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).

88. 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, 3; CS-1039, 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).

89. As identified above, the '552 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 7:55-60.

90. 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. The best 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 reencrypt 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.

91. For example, on its website[3] and in its publicly accessible data sheet[4], 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/sourcefirest

a network administrator could install and use to protect a network along with, for example, a 3D Sensor with IPS (See Section X.A.1 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.

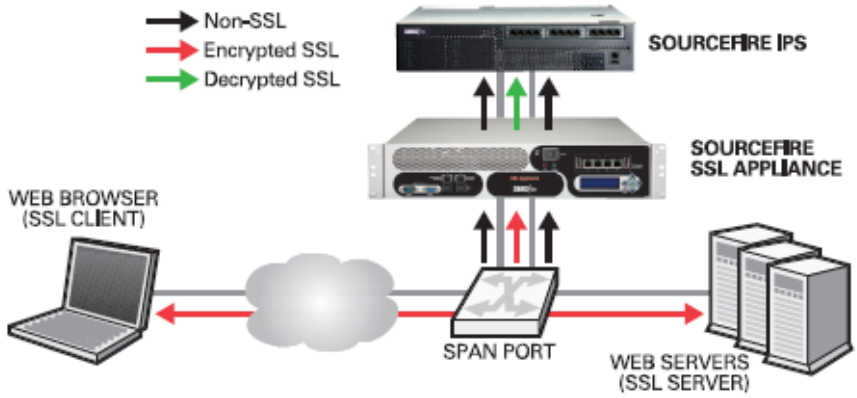


Figure 1. Passive IDS Configuration

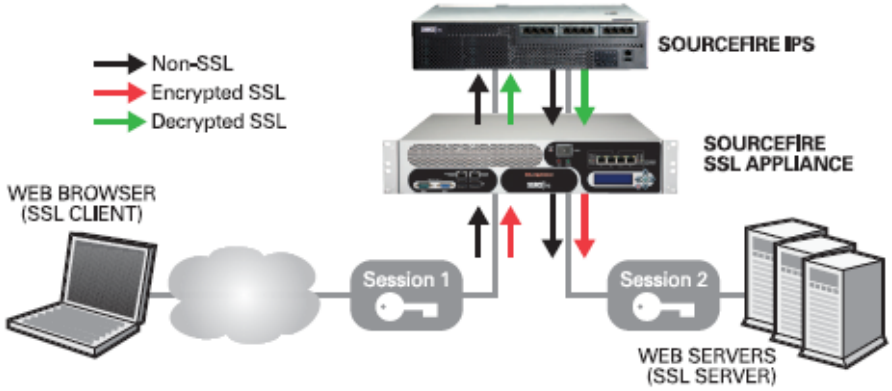


Figure 2. Inline IPS Configuration

CS-1010; CS-1011.

92. Based on the above-described historical development of the network security systems art, my opinion is that the claimed features of the ’552 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 employed a two-stage packet filtering process that used customizable rules to filter packets using the 5-tuple

(Stage I), as well as filtering based on application header information (Stage II), or indeed any aspect of application data.

93. In this declaration, I have analyzed the claims of the ’552 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 header information including PUT and GET commands, and TLS version to block or allow the packets (Stage II).

VIII. Claim Construction

94. I have been informed and understand that in order to properly evaluate the ’552 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.

95. In order to construe the claims, I have reviewed the entirety of the ’552 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

96. In my opinion, the Challenged Claims of the ’552 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.

97. Sourcefire describes a two-stage packet filtering process that included a first stage filtering based on the packet header 5-tuple, as well as a second stage filtering based on “keywords” which can include application layer data such as encryption protocol version (*e.g.* TLS 1.0, SSL 2.0) and HTTP methods such as GET and PUT.

98. 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. Therefore, the Challenged Claims are obvious over Sourcefire in view of the knowledge of a POSA.

X. Challenge: Claims 1-21 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 1-21 in View of the Knowledge and Skill of a POSA

99. The Sourcefire reference is a user manual titled "Sourcefire 3D System User Guide" Version 4.10 ("Sourcefire").

100. 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.

101. 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 '552 Patent.

1. Sourcefire

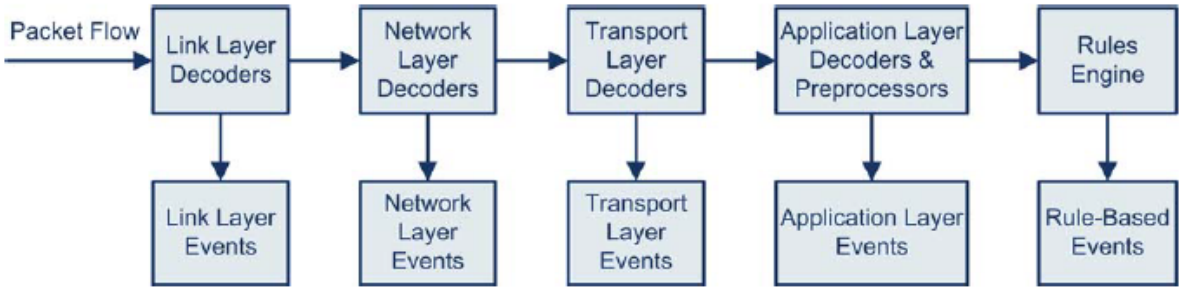
102. Sourcefire is a user manual for the Sourcefire 3D System. Sourcefire describes that the Sourcefire 3D System could identify changing assets and

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.

103. 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.

104. 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. 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.

105. 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.

106. 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.

107. 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. CS-1004 pp. 256-260, 428-430, 761-770.

108. 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.

109. 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.

110. 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 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 system could more easily identify traffic that indicated attacks and policy evasions. CS-1004 pp. 403-406, 435-439, 485, 762-776.

111. 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 rules[5] contained two logical parts: the rule header and the rule options.

112. 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.

113. 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.

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

Rule Header

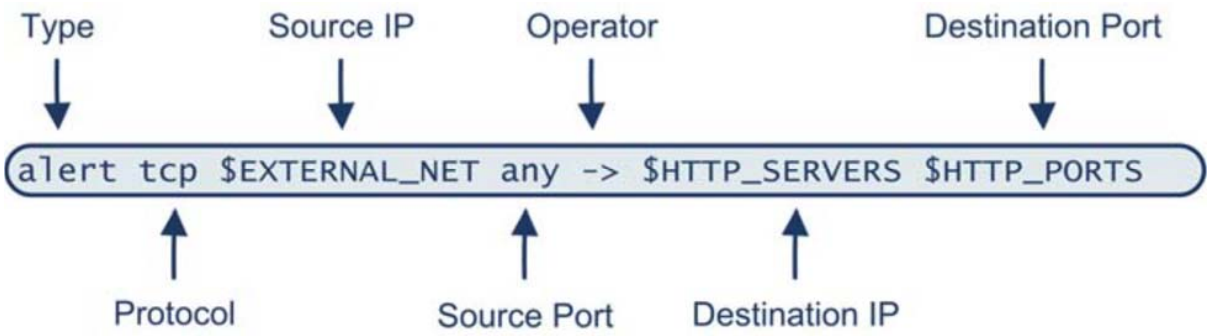
```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS $HTTP_PORTS
```

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.

115. 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.

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

CS-1004 pp. 764.

117. 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 disclosed that the rules engine and preprocessors allowed the combination of any number of compatible keywords to create highly specific intrusion rules. 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.

118. Thus, 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 IPS.

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

application layer header (*e.g.*, SSL/TLS Record header) of received packets and/or unencrypted application layer payload (*e.g.*, unencrypted SSL/TLS Record) of received handshake message[6] packets for an encrypted session:

`ssl_version` **Arguments**

Argument	Purpose
`sslv2`	Matches against traffic encoded using Secure Sockets Layer (SSL) Version 2.
`sslv3`	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.

120. When an intrusion rule used the ssl_version keyword, the rules engine invoked the SSL preprocessor to parse SSL/TLS traffic and find version

[6] As I discussed above, the SSL/TLS Record of "Finished" handshake message packets is encrypted. Section VII.D, ¶¶83-88.

information. 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.D, ¶85; 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 Record boundaries and extract the information in the Record headers). Thus, a POSA would know 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.

121. Thus, a user could protect his or her network against a security vulnerability associated with a specific encryption protocol version (*e.g.*, SSL v2 or TLS v1.0) (see Section VII.D ¶¶81-82) by writing a rule using the ssl_version keyword to invoke the SSL preprocessor to identify traffic using that version of SSL or TLS, and either *pass* (*i.e.*, allow), generate an *alert* (and pass), or *drop* (*i.e.*, block) packets identifying such a protocol version. Sourcefire explicitly discloses that such rules could be implemented by the rules engine invoking the SSL preprocessor to inspect the SSL/TLS traffic. CS-1004 pp. 827-828, 491, 597-601,

435-439, 697-700. As I discussed above, a POSA understood that SSL/TLS handshake message packets also included unencrypted or encrypted SSL/TLS Records. CS-1015; CS-1038; CS-1039; Section VII.D, ¶¶83-88.

122. Sourcefire also discloses that HTTP content keywords of preprocessor rules and intrusion rules 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." CS-1004 pp. 785-786, 700.

123. A POSA understood that identifying HTTP methods was important in existing security applications 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), and HTTP methods were used in IDS rules at least as early as 1997. CS-1028 pp. 4-80.

124. A POSA reading Sourcefire understood that a user could protect his or her network against such 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.*, "PUT"), and either *pass* (*i.e.*, allow), generate an *alert* (and pass), or *drop* (*i.e.*, block) packets identifying such an HTTP method in their application layer

payload content. Sourcefire discloses that such rules can be implemented by the HTTP inspect preprocessor and by the rules engine and provides a specific keyword option just to access the HTTP method. CS-1004 pp. 785-786, 807, 435-439, 491.

125. In summary, Sourcefire describes filtering packets based on packet header information including the 5-tuple, just like the Stage I evaluation described in the ’552 Patent. Every intrusion rule is qualified by the 5-tuple, and Sourcefire describes that, for a given 3D Sensor with IPS, this Stage I evaluation is performed as a packet arrives at the network layer decoder looking at IP address and protocol, and the transport layer decoder looking at ports. CS 1004 pp. 256-259, 764-769.

126. Sourcefire discloses that the rules can be applied on a packet by packet basis. CS-1004 p. 259. Sourcefire also describes how a user could design various rules to *pass* (and ignore), pass (and *alert*), and/or *drop* (*i.e.* block) packets based on header information including SSL/TLS version in application layer SSL/TLS Record headers and application layer payload information such as HTTP commands, just like the Stage II evaluation described in the ’552 Patent. For example, Sourcefire describes that, for a given 3D Sensor with IPS, this Stage II evaluation is performed as a packet arrives at various preprocessors and at the rules engine based on the intrusion rules. CS-1004 pp. 256-260.

B. Detailed Invalidity Analysis

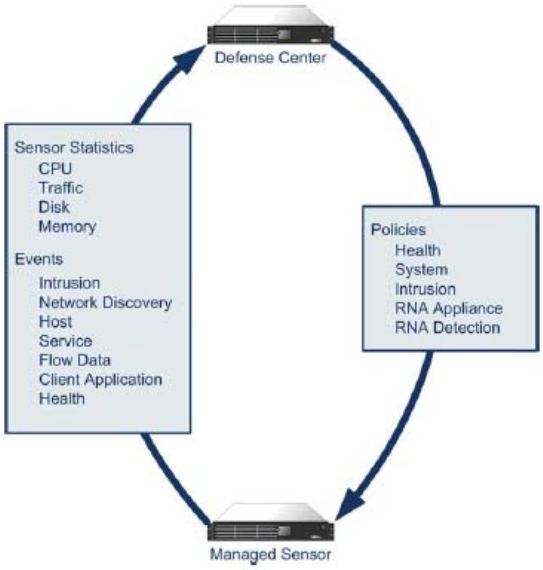
127. As detailed below, in my opinion, Sourcefire in view of the knowledge, skill and creativity of a POSA discloses every element of Claims 1-21. Each of the limitations of these claims are disclosed by Sourcefire and any potential differences that may be argued by the Patent Owner are implementation details that are well-within the knowledge and capability of a POSA.

Claim 1: *[1.0][7] A method, comprising: at a computing device comprising at least one processor, a memory, and a communication interface:*

128. Sourcefire discloses this limitation.

129. Sourcefire describes installation of 3D Sensors with IPS to monitor a network. Each 3D Sensor included a processor (CPU), memory, and disk storage, and, if managed by the centralized management service called the Defense Center, periodically sent statistics regarding such components (and events generated by applying rules to packets received via a communication interface) to such Defense Center.

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



CS-1004 pp. 33-34, 106-107.

130. Sourcefire discloses that each 3D Sensor received packets through a communication interface, *e.g.*, "The Sourcefire 3D System supports three types of interface sets . . . [including] inline . . . [and] inline with fail-open." CS-1004 p. 222-230. Therefore, each managed 3D Sensor is the claimed computer device having a processor, a memory, and a communication interface.

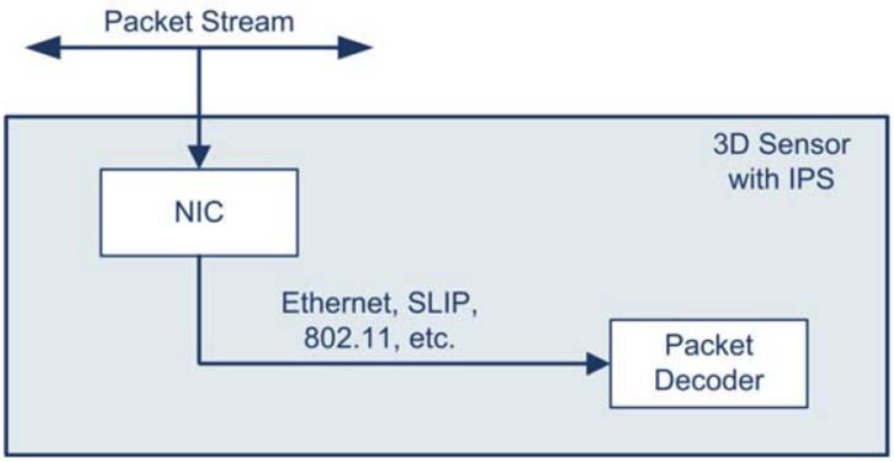
131. Therefore, Sourcefire discloses this limitation.

[1.2] receiving, via the communication interface, a plurality of hypertext transfer protocol secure (HTTPS) packets

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

133. Sourcefire discloses that two of the 3D Sensor's 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 communication 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.

134. Sourcefire discloses that rule headers in every intrusion rule specified 5-tuple information including the source and destination ports of each packet received by the rules engine of the 3D Sensor.

Source/Destination Port Syntax

To Specify...	Use	Example
any port	any	any
a specific port	the port number	80
a range of ports	a dash between the first and last port number in the range	80–443

CS-1004 pp. 768, 256-259, 600, 575, 761-764.

135. As shown in the example above, Sourcefire provides TCP port 443 as an example, which it describes as the destination port for HyperText Transfer Protocol Secure, or HTTPS. Sourcefire describes that the 5-tuple information specified in the rule header of an intrusion rule implemented by the network layer and transport layer decoders, SSL preprocessor, and/or rules engine could include destination port 443. CS-1004 pp. 768-769, 256, 600. Furthermore, Sourcefire discloses a preprocessor module specifically intended for dedicated processing of SSL/TLS traffic, the SSL preprocessor. CS-1004 pp. 596-601.

136. Therefore, Sourcefire discloses receiving HTTPS packets at the link layer decoder, SSL preprocessor, and rules engine via the communications interface (*e.g.*, inline interface) of the 3D Sensor.

[1.3] responsive to a determination by the at least one processor that at least a portion of the plurality of HTTPS packets have packet-header-field values corresponding to a packet filtering rule stored in the memory

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

138. Sourcefire discloses that rule headers in every intrusion rule specified 5-tuple information including the protocol (*e.g.*, TCP), source and destination IP addresses, source and destination ports (*e.g.*, 443)), the rule's action or type (*e.g.*, alert, drop, pass); and direction indicators showing the flow of traffic from source

to destination. CS-1004 pp. 762-769, 358-359. A POSA understands that the rules used by the 3D Sensor were stored in a memory accessed by the 3D Sensor.

139. Sourcefire provides an example of the determinations that were made from packet-header-field values (*e.g.*, destination port) in analyzed packets corresponding to the rule header of a packet filtering rule, stored in the 3D Sensor, and implemented by a preprocessor and/or rules engine of the 3D Sensor:

Rule Header Values

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

CS-1004 p. 764-765.

140. Sourcefire describes that the packet-header-field values specified in the rule header of an intrusion rule could include destination port 443. CS-1004 pp. 768-769, 256, 600. For example, a POSA understood that the network layer and transport layer decoders, SSL preprocessor and/or rules engine implementing such a rule would determine that packet header field values of at least a portion of

the received packets identified destination port 443 if such portion of the received packets were HTTPS packets.

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

[1.4] applying, by the at least one processor, an operator specified by the packet-filtering rule to the at least a portion of the plurality of HTTPS packets, wherein the operator specifies one or more application-header-field-value criteria identifying one or more transport layer security (TLS)-version values for which packets should be blocked from continuing toward their respective destinations

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

143. As I discussed above, a POSA understood that Sourcefire describes the use of SSL/TLS rule keywords to invoke the application layer SSL preprocessor and extract information about SSL or TLS version and session state from SSL/TLS Record headers in packets for an encrypted session. CS-1004 pp. 825-828, 491, 597-601, 697-700; Section X.A.1, ¶¶119-120. As I discussed above (Section VII.D), a POSA understood that SSL was a widely adopted standard technology[8] for keeping an internet connection secure and safeguarding any

[8] A POSA understood SSL version 2.0 was published in 1995 and TLS version 1.0 was published as a newer version of SSL version 3.0 in 1999. Section VII.D, ¶81.

sensitive data that is being sent between two systems. A POSA also understood that TLS was widely adopted as an updated, more secure, version of SSL. See Section VII.D. Sourcefire discloses that the SSL preprocessor (including when invoked by the rules engine) implemented the SSL preprocessor rules and intrusion rules including SSL keywords (*e.g.*, ssl_version) for both SSL and TLS. CS-1004 pp. 827-828, 491, 596-601, 700.

144. When a client and server communicate to establish an encrypted session using SSL or TLS, they exchange handshake messages to negotiate SSL/TLS parameters including the version of the SSL/TLS encryption protocol that the client and server will use during the "Finished" handshake messages and the rest of the encrypted session. CS-1004 p. 825; Section VII.D, ¶¶83-88. A POSA understood that, like the packets exchanged during the encrypted session, the packets including SSL/TLS Records for the handshake messages also included unencrypted application layer headers with a SSL/TLS version (*i.e.* SSL/TLS Record header). Section VII.D, ¶¶83-88. However, unlike the payload (*i.e.* SSL/TLS Record) of the encrypted session packets, and of the "Finished" handshake message packets, which is encrypted, the SSL/TLS Record of the other handshake message packets is not. CS-1004 p. 825; Section VII.D, ¶¶83-88.

145. Sourcefire explains that a user could configure the IPS running on a 3D Sensor to stop processing on all SSL/TLS Record Protocol packets in an

encrypted SSL/TLS session once encryption is established (*i.e.* including and after the "Finished" handshake messages). Sourcefire also explains that the default configuration of the IPS running on a 3D Sensor is to continue to inspect the SSL/TLS Record Protocol packets even if the SSL/TLS Records are encrypted. For the SSL/TLS Record Protocol packets, the application-layer SSL preprocessor (including when invoked by the rules engine) verified the 5-tuple information and other fields in the SSL/TLS Record headers based on the preprocessor and intrusion rules stored on the 3D Sensor. CS-1004 p. 597, 700.

146. Sourcefire describes that a user could configure SSL preprocessor rules and intrusion rules to only look for packets traveling over standard SSL/TLS ports (*e.g.*, TCP port 443 for HyperText Transfer Protocol Secure (HTTPS)) or could configure such rules to be "adaptive" to identify SSL/TLS Record Protocol packets traveling over non-standard ports. With adaptive profiles enabled in such rules, a POSA reading Sourcefire understood that the SSL preprocessor and/or rules engine checks the unencrypted application-layer header (*e.g.* SSL/TLS Record header) of packets for SSL/TLS identifiers to determine if the packets were SSL/TLS traffic. If a SSL/TLS identifier is found, the SSL preprocessor was invoked to process the now-identified SSL/TLS Record Protocol packets using the SSL keyword(s) and arguments of the preprocessor rules and intrusion rules even if the packets came over a non-standard SSL/TLS port. CS-1004 pp. 598, 697-701.

147. Sourcefire describes that keywords of intrusion rules could be used by the SSL preprocessor (when invoked by the rules engine) of a 3D Sensor to filter packets by encryption protocol version (*e.g.* TLS version or TLS's predecessor SSL version). Sourcefire discloses that that the keyword ssl_version could be included in such intrusion rules and used to block harmful SSL/TLS Record Protocol packets or allow benign SSL/TLS Record Protocol packets. CS-1004 pp. 827-828, 491, 597-601, 435-439.

148. For example, Sourcefire describes that the ssl_version keyword could be used by the application layer SSL preprocessor (when invoked by the rules engine) to match against such protocol version information of received packets including unencrypted and encrypted Records exchanged in negotiating an encrypted SSL/TLS session:

`ssl_version` **Arguments**

Argument	Purpose
`sslv2`	Matches against traffic encoded using Secure Sockets Layer (SSL) Version 2.
`sslv3`	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.

149. Therefore, Sourcefire discloses the use of the ssl-version keyword for writing intrusion rules for the rules engine (which will invoke the application layer SSL preprocessor) to identify the SSL/TLS version of received packets. As I discussed above, it would be obvious to a POSA that, for traffic in versions of SSL/TLS later than SSL version 2 (SSL v3, TLS 1.0-TLS 1.2), that the version could be obtained from the SSL/TLS Record header of SSL/TLS Record Protocol packets and that the application layer SSL preprocessor must look at the SSL/TLS Record headers in order to parse the SSL/TLS packets at all. Section X.A.1, ¶¶119-120.

150. Thus, a POSA understood that Sourcefire taught the use of ssl_version as a keyword, and thus it could be used as an application layer header field value in a packet filtering rule. Specifically, a POSA understood that the SSL/TLS version in an application layer SSL/TLS Record Protocol packet is not encrypted (*see* CS-

1001 7:53-60) and thus the 3D Sensor would be able to treat the SSL/TLS version just like the other fields in the application layer SSL/TLS Record header of such a packet and the packet header information contained in a 5-tuple. See also Section VII.D, ¶¶83-88.

151. To the extent that the Patent Owner argues that the claim requires that intrusion rules using the ssl_version keyword must inspect the SSL/TLS version values in SSL/TLS Record headers for encrypted *application-data* message packets (See also Section VII.D, ¶¶83-88), as opposed to the identical values in the identical SSL/TLS Record headers for encrypted and unencrypted *handshake* message packets (See also *id*.), then a POSA would have known how to write an intrusion rule so that the rules engine would invoke the SSL preprocessor to do so. The '552 Patent presumes that this is the case because it does not provide any examples. 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 SSL/TLS Record Protocol packets traveling over non-standard ports. See Section X.B, ¶146 above.

152. Further, Sourcefire describes that SSL v2 may have vulnerabilities associated with it and thus a POSA would be motivated to identify and block

packets including handshake message Records that proposed using SSL v2 for an SSL session[9]. CS-1004 pp. 827, 358-361, 435-439, 761-762.

153. Additionally, Sourcefire describes adding new rules as new vulnerabilities are discovered. CS-1004 p. 869. Sourcefire also discloses the procedure for writing new custom intrusion rules and adding them to the system. CS-1004 pp. 861-864. The BEAST attack tool distributed in 2011 demonstrated that TLS 1.0 encryption between browser and webserver could be readily broken in practice. CS-1037; CS-1038; CS-1039 p. 4; CS-1013; CS-1014; CS-1015. Therefore, because security vulnerabilities associated with TLS version 1.0 were widely known, a POSA in 2013 would be motivated to identify and block TLS Record Protocol packets including HTTP *application-data* message Records that were encrypted using TLS version 1.0 and TLS Record Protocol packets including handshake message Records that proposed using (or did use) TLS version 1.0 for a TLS session. CS-1004 pp. 254, 435-439, 761-762, 697-701; CS-1001 6:44-47; Section VII.D, ¶¶83-88. A POSA understood that by using the ssl_version keyword, packet filtering rules could be written to either *pass* or *drop* (*i.e.* block) the associated packets whose SSL/TLS version matched the keyword as taught by

---

[9] Security vulnerabilities with SSL version 2 were also widely known. CS-1037; CS-1039; CS-1016.

Sourcefire, and that doing so would have the predictable benefit of achieving increased network security by protecting a network against known security vulnerabilities.

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

[1.5] responsive to a determination by the at least one processor that one or more packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be blocked from continuing toward their respective destinations

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

156. As I discussed above, Sourcefire describes that part of an intrusion rule was a specified set of keywords and arguments. On a 3D Sensor with IPS, and implemented by the rules engine or a preprocessor, such an intrusion rule would detect attempts to exploit vulnerabilities on a network by analyzing network traffic to check if it matched the criteria in this part of the rule. The rules engine and applicable preprocessors of the IPS compared packets received via the inline communication interface against the conditions specified in each rule and, if the packet data matched all the conditions specified in a rule, the rule triggered. When a drop rule was triggered, the IPS dropped the packet and generated an event. CS-1004 pp. 761, 254, 262-264, 276, 358-361, 435-439.

157. As I discussed above, Sourcefire discloses packet filtering rules that used the ssl_version keyword in a rule to identify those packets that may present a vulnerability to the system. Sourcefire describes that packets presenting such a vulnerability could be dropped. In other words, the packets were blocked from continuing toward their respective destinations.

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

[1.6] applying, by the at least one processor, at least one packet-transformation function specified by the operator to the one or more packets to block each packet of the one or more packets from continuing toward its respective destination.

159. Sourcefire discloses this limitation.

160. Sourcefire describes that the 3D Sensor with IPS received the packets via an inline communication interface and could *pass* (*i.e.* allow), modify, or *block* (*i.e.* drop) packets. See Claim [1.2], [1.5]. The '552 Patent describes that passing or blocking transmission of the packet is a packet transformation function. CS-1001 9:26-40. Therefore, a POSA understood that Sourcefire discloses that the TLS version value for a packet could be used to apply a packet transformation function (block or drop) to block the packet from continuing toward its destination.

161. Therefore, Sourcefire discloses this limitation.

Claim 2: *[2.0] The method of claim 1, wherein the one more application-header-field-value criteria identify one or more TLS-version values for which packets*

should be allowed to continue toward their respective destinations, the method comprising,

162. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation. See also Claim [1.0]-[1.6].

163. Sourcefire discloses that, for packets received by a 3D Sensor with IPS via an inline communication interface, a user could configure the rule state of decoder rules, preprocessor rules, and intrusion rules to *pass* (and ignore), *alert* (and pass), and *drop* (*i.e.* block). If a rule was an *alert* rule, it generated an intrusion event but still allowed the traffic to pass to its destination. If a rule was a *pass rule*, it ignored the traffic allowing it to pass to its destination. CS-1004 pp. 761, 254, 276.

164. As I discussed above, Sourcefire describes that ssl_version keyword could be used to write rules, and thus a POSA understood that Sourcefire describes how to write a *pass* rule using the ssl_version keyword. See Claim [1.4]. As I also discussed above, it was widely known that certain SSL/TLS versions had security vulnerabilities (*i.e.*, SSL version 2, TLS version 1.0) and that the other TLS versions (*i.e.*, TLS version 1.1, 1.2) were more secure. See Claim [1.4].

165. A POSA understood that by using the ssl_version keyword (allowing values of tls1.1 and tls1.2), including in the "adaptive" mode, packet filtering rules could be written to have the application layer SSL preprocessor pass the associated TLS Record Protocol packets whose TLS version matched the keyword as taught

by Sourcefire, and that doing so would have the predictable benefit of achieving increased network security by protecting a network against known security vulnerabilities. See also Claim [1.4] above.

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

[2.1] responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be allowed to continue toward their respective destinations, applying, by the at least one processor, a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.

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

168. Sourcefire discloses that the keyword ssl_version can be used to filter packets that are received. For example, if a user wanted to identify any encrypted application data message or handshake "Finished" message packets that were not using, or other handshake message packets that were not proposing to use, SSL version 2.0, 3.0, or TLS version 1.0 (*i.e.*, not using SSL or TLS versions with known security vulnerabilities), the user could add ssl_version: (tls1.1, tls1.2) to a rule, including in "adaptive" mode. The application layer SSL preprocessor applying the rule would evaluate any TLS Record Protocol packets using TLS

version 1.1, or TLS version 1.2, and allow such packets to pass. CS-1004 p. 827, 697-701.

169. Therefore, a POSA understood that Sourcefire discloses that the TLS version value for a TLS Record Protocol packet would be used to apply a packet transformation function (pass) to allow a packet to continue toward its destination, and a POSA would be motivated to write an intrusion rule to do so, including in the "adaptive" mode, because it would have the predictable benefit of achieving increased network security by protecting a network against known security vulnerabilities while still permitting secure encrypted sessions to be established and maintained.

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

Claim 3: *[3.0] The method of claim 1, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be allowed to continue toward their respective destinations, the method comprising,*

171. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation. See also Claim [1.0]-[1.6].

172. Sourcefire discloses that the HTTP inspect preprocessor was an application layer preprocessor that could inspect all packets for traffic to and from web servers. CS-1004 p. 560. Sourcefire discloses that rules could be written

which included the most common HTTP methods of GET, PUT, POST and CONNECT as one or more of the rule criteria. CS-1004 pp. 568, 786.

173. 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), and HTTP methods were used in IDS rules at least as early as 1997. CS-1028 pp. 4-80; CS-1004 pp. 785-786, 807, 435-439, 491. Therefore, a POSA understood Sourcefire disclosed how a user would have written a rule using the HTTP GET Method option of the HTTP content keyword as part of the application layer rule criteria to invoke the HTTP inspect preprocessor to identify a packet using the "GET" HTTP method and to allow such a packet to continue towards its destination. See also Section X.A.1 ¶¶122-124.

174. As acknowledged by the '552 Patent, there is no technological difference between using HTTP method criteria over other types of application header rule criteria (*e.g.,* TLS version value, messaging protocol), and there are no specific technology changes needed to accommodate one application layer data protocol (*e.g.*, HTTP data transfer) over another (*e.g.*, messaging, encryption), or one HTTP method (*e.g.*, GET) over another (*e.g.*, PUT, CONNECT, POST). See also Section VI.A above. Thus, it is simply a design choice to select from a finite

number of identified, application header rule criteria that are relevant to a selected network vulnerability (*e.g.*, HTTP PUT method and certain application payload 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.

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

[3.1] responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be allowed to continue toward their respective destinations, applying, by the at least one processor, a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.

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

177. As I discussed above, Sourcefire discloses that the 3D Sensor with IPS will include a HTTP inspect preprocessor which can identify HTTP methods such as GET in a packet. See Claim [3.0]. Therefore, a POSA understood that Sourcefire discloses that the application-header-field values corresponding to the HTTP method of GET would be used to apply a packet transformation function (pass) to allow a GET packet to continue toward its destination.

178. To the extent that the Patent Owner asserts that the claim requires decrypting the HTTPS packets in order to identify the HTTP methods contained in the HTTPS packet, and that Sourcefire does not disclose this, such decryption method was known and could be readily applied by a POSA.

179. For example, the ’552 Patent includes no description of decryption of HTTPS packets in order to identify the HTTP methods. Specifically, FIG. 2 of the ’552 Patent illustrating the exemplary packet security gateway, includes no identification of encryption/decryption of packets. CS-1001, FIG. 2. The only description in the specification of the ’552 Patent of packet filtering of HTTPS packets is tied to the use of the TLS version number, which the patent points out “may not be encrypted.” CS-1001 7:58-60. Therefore, the specification of the ’552 Patent does not include a description of decryption of the HTTPS packets prior to or as part of packet filtering. Likewise, the ’552 Patent’s description of HTTP methods (including GET, PUT, POST CONNECT) are all tied to HTTP packets, not HTTPS packets. *See* CS-1001 2:21-39; 6:48-67; 7:30-48; FIG. 3 ((Rule 6 (Destination Port 80)), (Rule 5 (Destination Port 443)).

180. Therefore, to the extent that Patent Owner asserts that any of the claims require decryption of a HTTPS packet, that step would be obvious and well-

known as evidenced by the lack of description of any decryption methodologies in the ’552 Patent.[10].

181. For example, a POSA understood that a common technique to inspect and apply rules to encrypted application layer data in HTTPS packets was to have a packet-filtering device (or device serving the packet-filtering device) (1) serve as a proxy for the SSL/TLS handshake protocol between the client and server, (2) apply packet filtering rules to the packet header data and application header field values 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 (*e.g.*, common HTTP methods) and apply rules to either block or allow packets based on such rules, and then re-encrypt such data for transmission to its eventual destination. 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; CS-1010; CS-1011; Section VII.D.

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

---

[10]My opinion would apply equally to any claim that the Patent Owner asserts requires decryption of a HTTPS packet.

Claim 4: *[4.0] The method of claim 1, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be blocked from continuing toward their respective destinations, the method comprising*

183. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation. See also Claim [1.0]-[1.6].

184. Sourcefire identifies that the HTTP inspect preprocessor was an application layer preprocessor that could inspect all packets for traffic to and from web servers. CS-1004 pp. 560-561. Sourcefire discloses that rules could be written which included common HTTP methods of GET, PUT, POST and CONNECT as one or more of the rule criteria. CS-1004 pp. 568, 786. As I discussed above, 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," and HTTP methods were used in IDS rules at least as early as 1997. See Claim [3.0]. 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 application layer rule criteria to invoke the HTTP inspect preprocessor to identify a packet using the "PUT" HTTP method and to block such a packet with certain application payload content posing a threat from continuing towards its destination.

185. As acknowledged by the ’552 Patent, there is no technological difference between using HTTP method criteria over other types of application header rule criteria (*e.g.,* TLS version value, messaging protocol), and there are no specific technology changes needed to accommodate one application layer data protocol (*e.g.*, HTTP data transfer) over another (*e.g.*, messaging, encryption), or one HTTP method (*e.g.*, PUT) over another (*e.g.*, GET, CONNECT, POST). See also Section VI.A; Claim [3.0] above. Thus, it is simply a design choice to select from a finite number of identified, application header rule criteria that are relevant to a selected network vulnerability (*e.g.*, HTTP PUT method and certain application payload 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.

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

[4.1] responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be blocked from continuing toward their respective destinations applying, by the at least one processor, a packet-transformation function specified by the operator to the one or more other packets to block each packet of the one or more other packets from continuing toward its respective destination.

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

188. As I discussed above in Claim 3, Sourcefire discloses that the 3D Sensor with IPS included a HTTP inspect preprocessor which could identify HTTP methods such as PUT in a packet. See also Claim [3.0]-[3.1]. Therefore, a POSA understood that Sourcefire discloses that the application-header-field values corresponding to the HTTP method of PUT would have been used to apply a packet transformation function (drop) to block a PUT packet from continuing toward its destination. Therefore, Sourcefire discloses this limitation for the same reason as I discussed with respect to Claim 3, with the exception that the HTTP method value for which packet should be blocked is "PUT."

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

Claim 5: *[5.0] The method of claim 1, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods, and wherein the at least a portion of the plurality of HTTPS packets comprises a first portion of packets and a second portion of packets, each packet of the first portion comprising at least one GET method value, and each packet of the second portion comprising at least one of a PUT method value, a POST method value, or a CONNECT method value, the method comprising:*

190. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation for the same reason as I discussed with respect to Claim 3

when the HTTP method is GET, and with respect to Claim 4 when the HTTP method is PUT. See Claims [3.0]-[3.1], [4.0-[4.1]; See also Claim [1.0]-[1.6].

[5.1] allowing each packet of the first portion of packets to continue toward its respective destination; and

191. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation for the same reason as I discussed with respect to Claim 3 when the HTTP method is GET. See Claim [3.0]-[3.1].

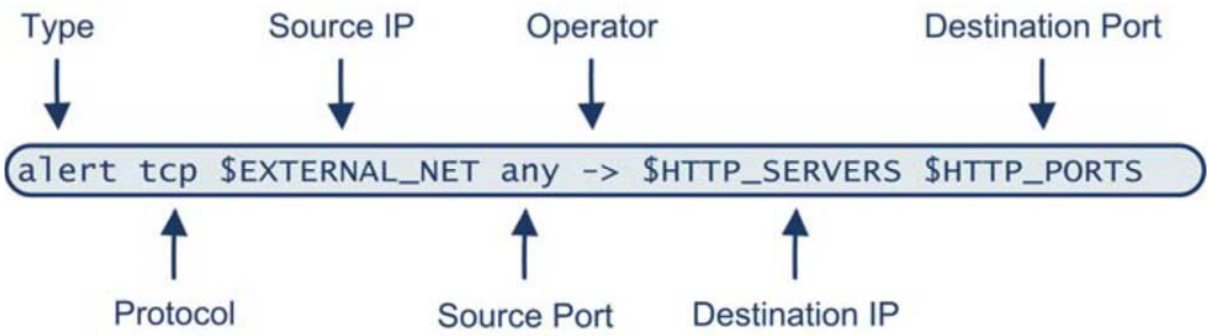
[5.2] blocking each packet of the second portion of packets from continuing toward its respective destination.

192. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation for the same reason as I discussed with respect to Claim 4 when the HTTP method is PUT. See Claim [4.0]-[4.1].

Claim 6: *[6.0] The method of claim 1, comprising, for each packet of the at least a portion of the plurality of HTTPS packets:*

193. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation. See also Claim [1.0]-[1.6].

194. Sourcefire discloses that packet filtering rules were used by the 3D Sensor to examine packets. Sourcefire describes that all of the intrusion rules had 5-tuple values associated with them. CS-1004 p. 256-259, 761-763. Sourcefire defines the information contained in the rule header of the packet filtering rule as shown below (including the 5-tuple):



CS-1004 p. 764.

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

Rule Header Values

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

CS-1004 p. 764.

196. Sourcefire explains that enabling a rule caused IPS to generate events on traffic matching the rule. CS-1004 p. 403. Therefore, the rule triggered when the step of comparing the rule header value with the packet header value of the packet received by the IPS produced a match.

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

[6.1] comparing a protocol of a data section of the packet to a protocol specified by the packet-filtering rule,

198. As shown in the table above, the rule header value included "Protocol" (*e.g.*, TCP) which IPS compared to the protocol of a data section of the packet. CS-1001 5:24-41, 9:2-8, 10:4-11; See Claim [6.0].

[6.2] comparing a source Internet Protocol (IP) address of the packet to one or more source IP addresses specified by the packet-filtering rule,

199. As shown in the table above, the rule header value included "Source IP Address." See Claim [6.0].

[6.3] comparing a source port of the packet to one or more source ports specified by the packet-filtering rule,

200. As shown in the table above, the rule header value included "Source Ports." See Claim [6.0].

[6.4] comparing a destination IP address of the packet to one or more destination IP addresses specified by the packet-filtering rule; and

201. As shown in the table above, the rule header value included "Destination IP Address." See Claim [6.0].

[6.5] comparing a destination port of the packet to one or more destination ports specified by the packet-filtering rule.

202. As shown in the table above, the rule header value included "Destination Ports." See Claim [6.0].

Claim 7: *[7.0] The method of claim 1, comprising: [7.0.1] receiving, via the communication interface, a second plurality of HTTPS packets; and*

203. Sourcefire in view of the knowledge, skill and creativity of a POSA discloses this limitation for the same reasons as I discussed with respect to Claim 1 for the first plurality of packets. See Claim [1.2].

[7.1] responsive to a determination by the at least one processor that the second plurality of HTTPS packets does not have packet-header-field values corresponding to the packet-filtering rule, blocking, by the computing device and without applying the operator to the second plurality of HTTPS packets, each packet of the second plurality of HTTPS packets from continuing toward its respective destination.

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

205. As I discussed above, Sourcefire discloses that, for packets received by a 3D Sensor via an inline communication interface, a user could configure the rule state of any decoder rule, preprocessor rule, and/or intrusion rule to "drop" which would block packets that triggered the rule. See Claim [1.2], [1.6].

206. Sourcefire discloses that rule headers of preprocessor and intrusion rules included, for example, the source and destination IP addresses of traffic that the rule inspected. Sourcefire discloses that the packet inspection could be restricted to the packets originating from specific IP addresses or those destined to a specific IP address in order to reduce the amount of packet inspection the system must perform. Sourcefire discloses the use of a "negate" indicator (denoted as "!") in the rule to trigger the rule for all packets from all IP addresses except the one associated with the negate indicator. CS-1004 p. 766, 761, 254, 276.

207. Therefore, in the chart below, Sourcefire discloses to a POSA that by specifying "!192.168.1.15" in a packet filtering rule, a user could block all IP packets from IP Source addresses other than 192.168.1.15." In other words, packets that did not have a IP Source Address (*a packet-header-field value*) corresponding to 192.168.1.15 (*the packet-filtering rule*), would be blocked from continuing toward their respective destination.

Source/Destination IP Address Syntax

To Specify...	Use...	Example
any IP address	any	any
a specific IP address	the IP address	192.168.1.1
a list of IP addresses	brackets ([]) to enclose the IP addresses and commas to separate them	[192.168.1.1,192.168.1.15]
a range of IP addresses	CIDR block notation	192.168.1.0/24
anything except a specific IP address or set of addresses	the ! character before the IP address or addresses you want to negate **IMPORTANT!** In the same rule, you cannot mix IP addresses that use the ! character with IP addresses that do not use it. You should create another rule.	!192.168.1.15

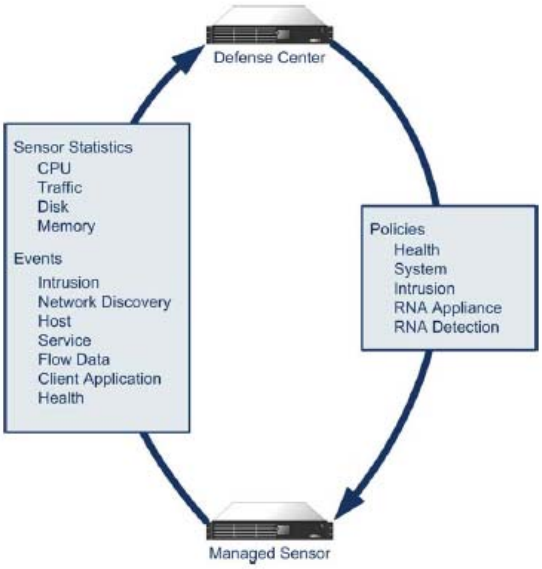
CS-1004 p. 767.

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

Claim 8: *[8.0] An apparatus, comprising: at least one processor; and a memory storing instructions that when executed by the at least one processor cause the apparatus to: receive a plurality of hypertext transfer protocol secure (HTTPS) packets; responsive to a determination by the at least one processor that at least a portion of the plurality of HTTPS packets have packet-header-field values corresponding to a packet-filtering rule, apply an operator specified by the packet-filtering rule to the at least a portion of the plurality of HTTPS packets, wherein the operator specifies one or more application-header-field-value criteria identifying one or more transport layer security (TLS)-version values for which packets should be blocked from continuing toward their respective destinations; and responsive to a determination by the at least one processor that one or more packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be blocked from continuing toward their respective destinations, apply at least one packet-transformation function specified by the operator to the one or more packets to*

block each packet of the one or more packets from continuing toward its respective destination.

209. Claim 8 is an apparatus claim corresponding to the method recited in Claim 1 and therefore each of the limitations recited in Claim 8 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as 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, and, if managed by the centralized management service called the Defense Center, periodically sent statistics regarding such components (and events generated by applying rules to packets received via a communication interface) to such Defense Center.



CS-1004 pp. 33-34, 106-107.

Claim 9: *[9.0] The apparatus of claim 8, wherein the one or more application-header-field-value criteria identify one or more TLS-version values for which packets should be allowed to continue toward their respective destinations, and wherein the instructions when executed by the at least one processor, cause the apparatus to, responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be allowed to continue toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.*

210. Claim 9 is an apparatus claim corresponding to the method recited in Claim 2 and therefore each of the limitations recited in Claim 9 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 2. See Claim [2.0]-[2.1].

Claim 10: *[10.0] The apparatus of claim 8, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be allowed to continue toward their respective destinations, and wherein the instructions, when executed by the at least one processor, cause the apparatus to, responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be allowed to continue toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.*

211. Claim 10 is an apparatus claim corresponding to the method recited in Claim 3 and therefore each of the limitations recited in Claim 10 are disclosed by

Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 3. See Claim [3.0]-[3.1].

Claim 11: *[11.0] The apparatus of claim 8, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be blocked from continuing toward their respective destinations, and wherein the instructions, when executed by the at least one processor, cause the apparatus to, responsive to a determination by the at least one processor that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be blocked from continuing toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to block each packet of the one or more other packets from continuing toward its respective destination.*

212. Claim 11 is an apparatus claim corresponding to the method recited in Claim 4 and therefore each of the limitations recited in Claim 11 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 4. See Claim [4.0]-[4.1].

Claim 12: *[12.0] The apparatus of claim 8, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods, wherein the at least a portion of the plurality of HTTPS packets comprises a first portion of packets and a second portion of packets, each packet of the first portion comprising at least one GET method value, and each packet of the second portion comprising at least one of a PUT method value, a POST method value, or a CONNECT method value, and wherein the instructions, when executed by the at least one processor, cause the apparatus to: allow each packet of the first portion to continue toward its respective destination; and block each packet of the second portion from continuing toward its respective destination.*

213. Claim 12 is an apparatus claim corresponding to the method recited in Claim 5 and therefore each of the limitation recited in Claim 12 are disclosed by

Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 5. See Claim [5.0]-[5.2].

Claim 13: *[13.0] The apparatus of claim 8, wherein the instructions, when executed by the at least one processor, cause the apparatus to, for each packet of the at least a portion of the plurality of HTTPS packets: compare a protocol of a data section of each of the packet to a protocol specified by the packet-filtering rule; compare a source Internet Protocol (IP) address of the packet to one or more source IP addresses specified by the packet-filtering rule; compare a source port of the packet to one or more source ports specified by the packet-filtering rule; compare a destination IP address of the packet to one or more destination IP addresses specified by the packet-filtering rule; and compare a destination port of the packet to one or more destination ports specified by the packet-filtering rule.*

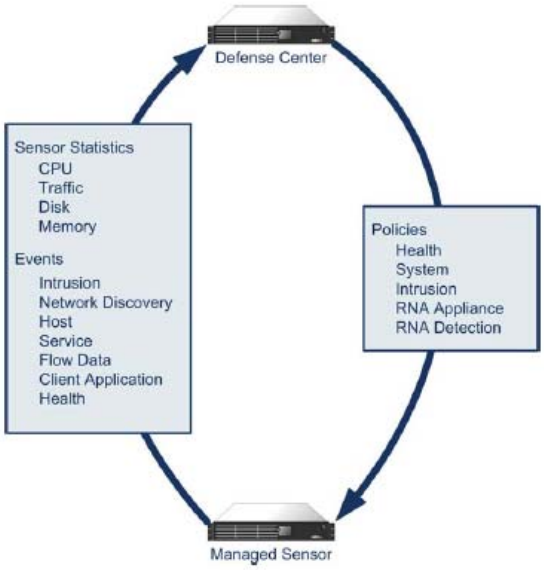
214. Claim 13 is an apparatus claim corresponding to the method recited in Claim 6 and therefore each of the limitation recited in Claim 13 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 6. See Claim [6.0]-[6.5].

Claim 14: *[14.0] The apparatus of claim 8, wherein the instructions, when executed by the at least one processor, cause the apparatus to: receive a second plurality of HTTPS packets; and responsive to a determination by the at least one processor that the second plurality of HTTPS packets does not have packet-header-field values corresponding to the packet-filtering rule, block, without applying the operator to the second plurality of HTTPS packets, each packet of the second plurality of HTTPS packets from continuing toward its respective destination.*

215. Claim 14 is an apparatus claim corresponding to the method recited in Claim 7 and therefore each of the limitation recited in Claim 14 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 7. See Claim [7.0]-[7.1].

Claim 15: *[15.0] One or more non-transitory computer-readable media having instructions stored thereon that when executed by one or more computing devices cause the one or more computing devices to: receive a plurality of hypertext transfer protocol secure (HTTPS) packets; responsive to a determination by the one or more computing devices that at least a portion of the plurality of HTTPS packets have packet-header-field values corresponding to a packet-filtering rule, apply an operator specified by the packet-filtering rule to the at least a portion of the plurality of HTTPS packets, wherein the operator specifies one or more application-header-field-value criteria identifying one or more transport layer security (TLS)-version values for which packets should be blocked from continuing toward their respective destinations; and responsive to a determination by the one or more computing devices that one or more packets, of the at least a portion of the plurality of HTTPS packets have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be blocked from continuing toward their respective destinations, apply at least one packet-transformation function specified by the operator to the one or more packets to block each packet of the one or more packets from continuing toward its respective destination.*

216. Claim 15 is a computer program claim corresponding to the method recited in Claim 1 and therefore each of the limitation recited in Claim 15 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as 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, and, if managed by the centralized management service called the Defense Center, periodically sent statistics regarding such components (and events generated by applying rules to packets received via a communication interface) to such Defense Center. 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 16: *[16.0] The one or more non-transitory computer-readable media of claim 15, wherein the one or more application-header-field-value criteria identify one or more TLS-version values for which packets should be allowed to continue toward their respective destinations, and wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to, responsive to a determination by the one or more computing devices that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more TLS-version values of the one or more TLS-version values for which packets should be allowed to continue toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.*

217. Claim 16 is a computer program claim corresponding to the method recited in Claim 2 and therefore each of the limitation recited in Claim 16 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 2. See Claim [2.0]-[2.1].

Claim 17: *[17.0] The one or more non-transitory computer-readable media of claim 15, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be allowed to continue toward their respective destinations, and wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to, responsive to a determination by the one or more computing devices that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be allowed to continue toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to allow each packet of the one or more other packets to continue toward its respective destination.*

218. Claim 17 is a computer program claim corresponding to the method recited in Claim 3 and therefore each of the limitation recited in Claim 17 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 3. See Claim [3.0]-[3.1].

Claim 18: *[18.0] The one or more non-transitory computer-readable media of claim 15, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods for which packets should be blocked from continuing toward their respective destinations, and wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to, responsive to a determination by the one or more computing devices that one or more other packets, of the at least a portion of the plurality of HTTPS packets, have one or more application-header-field values corresponding to one or more HTTP methods of the one or more HTTP methods for which packets should be blocked from continuing toward their respective destinations, apply a packet-transformation function specified by the operator to the one or more other packets to block each packet of the one or more other packets from continuing toward its respective destination.*

219. Claim 18 is a computer program claim corresponding to the method recited in Claim 4 and therefore each of the limitation recited in Claim 18 are

disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 4. See Claim [4.0]-[4.1].

Claim 19: *[19.0] The one or more non-transitory computer-readable media of claim 15, wherein the one or more application-header-field-value criteria identify one or more hypertext transfer protocol (HTTP) methods, wherein the at least a portion of the plurality of HTTPS packets comprises a first portion of packets and a second portion of packets, each packet of the first portion comprising at least one GET method value, and each packet of the second portion comprising at least one of a PUT method value, a POST method value, or CONNECT method value, and wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to: allow each packet of the first portion to continue toward its respective destination; and block each packet of the second portion from continuing toward its respective destination.*

220. Claim 19 is a computer program claim corresponding to the method recited in Claim 5 and therefore each of the limitation recited in Claim 19 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 5. See Claim [5.0]-[5.2].

Claim 20: *[20.0] The one or more non-transitory computer-readable media of claim 15, wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to, for each packet of the at least a portion of the plurality of HTTPS packets: compare a protocol of a data section of the packet to a protocol specified by the packet-filtering rule; compare a source Internet Protocol (IP) address of the packet to one or more source IP addresses specified by the packet-filtering rule; compare a source port of the packet to one or more source ports specified by the packet-filtering rule; compare a destination IP address of the packet to one or more destination IP addresses specified by the packet-filtering rule; and compare a destination port of the packet to one or more destination ports specified by the packet-filtering rule.*

221. Claim 20 is a computer program claim corresponding to the method recited in Claim 6 and therefore each of the limitation recited in Claim 20 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 6. See Claim [6.0]-[6.5].

Claim 21: *[21.0] The one or more non-transitory computer-readable media of claim 15, wherein the instructions, when executed by the one or more computing devices, cause the one or more computing devices to: receive a second plurality of HTTPS packets; and responsive to a determination by the one or more computing devices that the second plurality of HTTPS packets does not have packet-header field values corresponding to the packet-filtering rule, block, without applying the operator to the second plurality of HTTPS packets, each packet of the second plurality of HTTPS packets from continuing toward its respective destination.*

222. Claim 21 is a computer program claim corresponding to the method recited in Claim 7 and therefore each of the limitation recited in Claim 21 are disclosed by Sourcefire in view of the knowledge, skill, and creativity of a POSA for the same reasons as I discussed with respect to Claim 7. See Claim [7.0]-[7.1].

XI. Conclusion and Declaration

223. 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.

224. 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.

225. 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: July 20, 2018

By: /Dr. Stuart Staniford/
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 0wn 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 Phi^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