

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

CISCO SYSTEMS INC.,

Petitioner

IPR2025-00837

U.S. Patent No. 11,106,824

**DECLARATION OF NADER F. MIR, PH.D.,
UNDER 37 C.F.R. § 1.68 IN SUPPORT OF PETITION
FOR *INTER PARTES* REVIEW**

TABLE OF CONTENTS

I.	INTRODUCTION	4
II.	QUALIFICATIONS AND PROFESSIONAL EXPERIENCE	8
III.	LEVEL OF ORDINARY SKILL IN THE ART	11
IV.	RELEVANT LEGAL STANDARDS	13
V.	BACKGROUND	14
	A. TCP/IP and OSI Models.....	14
	B. Packet Encapsulation.....	19
	C. Encryption	21
	D. Intrusion Detection System (IDS).....	23
VI.	OVERVIEW OF THE '824 PATENT	25
VII.	CLAIM CONSTRUCTION	27
VIII.	IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE....	27
IX.	THE COMBINATION OF BURNS AND YANG RENDERS OBVIOUS CLAIMS 17-20.....	28
	A. Summary of Burns.....	28
	B. Summary of Yang	31
	C. Reasons to Combine Burns and Yang.....	33
	1. Claim 17	34
	2. Claim 18	59
	3. Claim 19	67
	4. Claim 20	69
X.	THE COMBINATION OF BURNS, YANG, AND WITTENBERG RENDERS OBVIOUS CLAIMS 1-16.....	69
	A. Summary of Wittenberg	69
	B. Reasons to Combine Wittenberg, Burns, and Yang.....	73
	C. Detailed Analysis of Claims.....	75
	1. Claim 1	75

2.	Claim 2.....	91
3.	Claim 3.....	92
4.	Claim 4.....	98
5.	Claim 5.....	99
6.	Claim 6.....	103
7.	Claim 7.....	110
8.	Claim 8.....	112
9.	Claim 9.....	114
10.	Claim 10.....	118
11.	Claim 11.....	120
12.	Claim 12.....	122
13.	Claim 13.....	122
14.	Claim 14.....	124
15.	Claim 15.....	126
16.	Claim 16.....	128
XI.	CONCLUSION.....	128

I, Nader F. Mir, Ph.D., do hereby declare as follows:

I. INTRODUCTION

1. I am making this declaration at the request of Cisco Systems, Inc. in the matter of the *Inter Partes* Review of U.S. Patent No. 11,106,824 (“the ’824 patent”) to Koren and Wasserman.

2. I am being compensated for my work in this matter at my standard hourly rate. I am also being reimbursed for reasonable and customary expenses associated with my work and testimony in this proceeding. My compensation is not contingent on the outcome of this matter or the specifics of my testimony.

3. I have been asked to provide my opinions regarding whether the subject matter of claims 1-20 (“the Challenged Claims”) of the ’824 patent would have been obvious to a person having ordinary skill in the art (“POSITA”) at the time of the alleged invention, in light of the prior art. It is my opinion that the Challenged Claims would have been obvious to a POSITA.

4. In the preparation of this declaration, I have studied:

- a. The ’824 patent, Ex.1001;
- b. The prosecution history of the ’824 patent (“’824 File History”), Ex.1002;
- c. U.S. Patent No. 8,341,724 to Burns and Sukhanov (“Burns”), Ex. 1005;
- d. U.S. Patent No. 8,291,495 to Burns and Yang (“Yang”), Ex.1006;

e. U.S. Patent Publication No. 2005/0078668 to Wittenberg et al. (“Wittenberg”), Ex.1007.

5. In forming the opinions expressed below, I have considered: the documents listed above; the relevant legal standards, including the standard for obviousness; and my own knowledge and experience based upon my work in the field of network communications and security as described below, as well as portions of the following additional materials:

- a. Transmission Control Protocol, RFC 793 (Sept. 1981), Ex.1011;
- b. Andrew Tanenbaum, “Computer Networks,” 3rd ed. (1996) (“Tanenbaum”), Ex.1012;
- c. U.S. Patent Appl. No. 2006/0215695 to Olderdissen (“Olderdissen”), Ex.1013;
- d. Perlman, Radia, “Interconnections: Bridges, Routers, Switches, and Internetworking Protocols,” 2d ed. (2000), Ex.1014;
- e. Kozierek, Charles M., “The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference,” (2005), Ex.1016;
- f. Nayak, Umesha; Hodeghatta Rao, Umesh, “The InfoSec Handbook: An Introduction to Information Security,” (2014), Ex.1017;
- g. U.S. Patent No. 7,406,522 to Riddle, Ex.1018;
- h. U.S. Patent Publication No. 2011/0106710 to Reed et al., Ex.1019;

- i. U.S. Patent Publication No. 2011/0022835 to Schibuk et al., Ex.1020;
- j. U.S. Patent No. 6,944,762 to Garrison, Ex.1021;
- k. U.S. Patent Publication No. 2005/0281192 to Nadeau et al., Ex.1022;
- l. U.S. Patent Publication No. 2011/0110328 to Pradeep et al., Ex.1023;
- m. U.S. Patent Publication No. 2013/0329578 to Groves et al., Ex.1024;
- n. U.S. Patent Publication No. 2002/0046264 to Dillon et al., Ex.1025;
- o. U.S. Patent Publication No. 2002/0059435 to Border et al., Ex.1026;
- p. U.S. Patent No. 6,826,620 to Mawhinney et al., Ex.1027;
- q. U.S. Patent Publication No. 2017/0230065 to Saraswathyama et al., Ex.1028;
- r. U.S. Patent Publication No. 2014/0351106 to Furr et al., Ex.1029;
- s. U.S. Patent Publication No. 2012/0230208 to Pyatkovskiy, Ex.1030;
- t. U.S. Patent Publication No. 2006/0248582 to Panjwani et al., Ex.1031;
- u. U.S. Patent Publication No. 2004/0013112 to Goldberg et al., Ex.1032;
- v. Transmission Control Protocol, RFC 1700 (1994), Ex.1033;
- w. Transmission Control Protocol, RFC 791 (1981), Ex.1034;
- x. Transmission Control Protocol, RFC 790 (1981), Ex.1035;
- y. Proctor, Paul E., “The Practical Intrusion Detection Handbook,” (2001),
Ex.1036;
- z. U.S. Patent Publication No. 2008/0159146 to Claudatos et al., Ex.1037;
- aa. U.S. Patent Publication No. 2007/0088845 to Memon et al., Ex.1038;

- bb. U.S. Patent Publication No. 2005/0047449 to Adolph et al., Ex.1039;
- cc. Transmission Control Protocol, RFC 5246 (2008), Ex.1040;
- dd. U.S. Patent Publication No. 2003/0038791 to Chou, Ex.1041;
- ee. U.S. Patent No. 5,289,589 to Bingham et al., Ex.1042;
- ff. U.S. Patent Publication No. 2006/0040650 to Schepers et al., Ex.1043;
- gg. U.S. Patent Publication No. 2003/0014624 to Maturana et al., Ex.1044;
- hh. U.S. Patent Publication No. 2006/0209858 to Blum, Ex.1045;
- ii. Transmission Control Protocol, RFC 4253 (2006), Ex.1046;
- jj. U.S. Patent Publication No. 2002/0143897 to Patil, Ex.1047;
- kk. U.S. Patent Publication No. 2007/0248105 to Shinoda et al., Ex.1048;
- ll. U.S. Patent Publication No. 2008/0016570 to Capalik, Ex.1049;
- mm. U.S. Patent No. 7,127,743 to Khanolkar et al., Ex.1050;
- nn. U.S. Patent Publication No. 2006/0179472 to Chang et al., Ex.1051;
- oo. U.S. Patent Publication No. 2004/0187028 to Perkins et al., Ex.1052;
- pp. U.S. Patent Publication No. 2009/0254970 to Agarwal et al., Ex.1053;
- qq. U.S. Patent Publication No. 2013/0179753 to Flynn et al., Ex.1054;
- rr. U.S. Patent Publication No. 2014/0207997 to Peterson et al., Ex.1055;
- ss. U.S. Patent Publication No. 2012/0303952 to Smith et al., Ex.1056.

6. Unless otherwise noted, all emphasis in any quoted material has been added. Claim terms are italicized.

II. QUALIFICATIONS AND PROFESSIONAL EXPERIENCE

7. My complete qualifications and professional experience are described in my *Curriculum Vitae*, a copy of which can be found in Exhibit 1004. The following is a brief summary of my relevant qualifications and professional experience.

8. I am a tenured professor in the Electrical Engineering Department of San Jose State University, California, where I also held the position of Associate Chair from 2006 through 2008. I have been a faculty member at San Jose State University since 2001.

9. In 1995, I received a Ph.D. in electrical engineering from Washington University, St. Louis. I received an M.S. degree in electrical engineering in 1990, also from Washington University. In 1985, I received a B.S. in electrical engineering from Polytechnic University.

10. I have over 30 years of experience with computer networks and communication systems, including both industry and academic experience. Before my current position at San Jose State University, I was an Assistant Professor in the Department of Electrical and Computer Engineering at the University of Kentucky. I joined San Jose State University in 2001 as an Associate Professor. Since 2005, I have been a Full Professor in the Electrical Engineering Department at San Jose State University, where I perform research and teach in the areas of computer

networks, networking devices, switches, routers, virtualization, virtual switches, cloud computing, software-defined networking in clouds, and cloud data centers.

11. I held several technical editorial positions for various journals, including IEEE Communication Magazine. As a Technical Editor of IEEE Communications Standards Magazine for Cloud and Edge Computing, I was responsible for accepting or rejecting scientific articles submitted to the journal in the areas of computer networking focusing on cloud systems. From 2005 through 2020, I held the technical editorial position for IEEE Communications Magazine. I am a senior member of the IEEE and have served as a member of the technical program committees and the steering committees for a number of major IEEE communications and networking conferences.

12. For more than 30 years, I have conducted research in the areas of computer and communication networks; networking devices; protocols including (but not limited to) packet-switched networks, cloud and edge computing, software-defined networking in cloud systems, integrated voice, video, data center networks, TCP/IP, client/server operations, content delivery networks, multimedia and streaming, client/server, networking traffic flow analysis, wireless/mobile networks, networking devices, network security, and network virtualization, among others.

13. I have received a number of university awards as well as national and international awards. In particular, I have received a number of research grants from

private, state, and governmental funding agencies for conducting research in the fields of computers and communication networks. I am also the recipient of several university teaching recognition awards from San Jose State University; several research excellence awards; and the school's published book award for the year.

14. Before my positions in academia, I worked in various positions in the telecommunication and computer network industries. From 1985 to 1988, I was a Telecomm R&D Engineer at the Telecommunication Research & Development Center (TRDC) in Surrey, England. In particular, I headed engineering projects in the area of telecommunication engineering. I also participated in the design of a high-speed switch for digital Private Branch Exchange (PBX) and SS7 networks. In 1991, I worked as an Electrical Engineering Consultant for Bussmann Company, where I instructed engineers on electrical engineering topics and prepared them for the National Principles and Practice of Engineering (PE) exam. From 1989 to 1995, I worked as a Computer/Communications Chip-Design Engineer at the Computer & Communications Research Center in St. Louis, Missouri. In that position, I designed a VLSI gigabit switch control plane chip for a high-speed computer network using CMOS technology. From 1995 to 1996, I was a Research Scientist Engineer at Advanced Telecommunications Institute, Stevens Institute of Technology in Hoboken, New Jersey. There I headed analyzing communications and high-speed computer networking projects.

15. Since obtaining my Ph.D., I have served in various consulting positions in the area of computer networking technology and communication engineering, including for CITS Inc. in San Jose, California (2012-2013), Morrison & Foerster LLP in San Francisco, California (2015-2016), and Butler and Snow in Jackson, Mississippi (2017-2018).

16. I have authored or co-authored over 100 articles in peer-reviewed journals, conference proceedings, texts, and monographs in the aforementioned areas of computer networking, communications, and related fields. I have published two textbooks, one of which is entitled “Computer Communication Networks.” This textbook has been adopted by numerous universities worldwide.

17. I am a named inventor on one U.S. patent, which relates to a packet-switching network and method of routing packets.

18. I have served as an expert witness and technical consultant in litigation and *inter partes* review matters concerning computer networks, wireless networks at both the protocol and hardware levels, and telecommunications.

III. LEVEL OF ORDINARY SKILL IN THE ART

19. I understand there are multiple factors relevant to determining the level of ordinary skill in the pertinent art, including (1) the levels of education and experience of persons working in the field at the time of the invention; (2) the

sophistication of the technology; (3) the types of problems encountered in the field; and (4) the prior art solutions to those problems.

20. A POSITA in the field of the '824 patent, as of its priority date of April 9, 2017, would have been someone knowledgeable and familiar with the network communications techniques and packet processing technologies available in the mid-2000s. The '824 patent “relates to data management,” in the context of packet-based “communication between a remote server and a user’s device.” '824 patent, 1:15, Abstract. A POSITA would have had a working knowledge of the data communications art that is pertinent to the '824 patent, including packet-based computer networking. *See* '824 patent, 6:20-23. The '824 patent refers to a variety of computer components, networking terms, and protocol acronyms without explanation, indicating that a POSITA would be familiar with a variety of computer and internet networking topics such as the World Wide Web and Transmission Control Protocol / Internet Protocol (TCP/IP). *See* '824 patent, 9:1-9, 9:67, 5:47-7:39. Such a POSITA would have a bachelor’s degree in computer science, computer engineering, electrical engineering, or equivalent training, and approximately three years of experience relating to packet-based network communications. Additional work experience can substitute for specific educational background, and vice versa.

21. For purposes of this Declaration, in general, and unless otherwise noted, my statements and opinions, such as those regarding my own experience and what a

POSITA would have understood or known generally (and specifically related to the references I consulted herein), reflect the knowledge that existed in the relevant field as of the priority date of the '824 patent.

IV. RELEVANT LEGAL STANDARDS

22. I am not an attorney. In preparing and expressing my opinions and considering the subject matter of the '824 patent, I am relying on certain basic legal principles that Cisco's counsel has explained to me.

23. I understand that prior art to the '824 patent includes patents and printed publications in the relevant art that predate the priority date of the '824 patent. For purposes of this Declaration, I am applying April 9, 2017, as the priority date of the '824 patent.

24. I have been informed by Cisco's counsel that a claimed invention is unpatentable under 35 U.S.C. §103 if the differences between the claimed invention and the prior art are such that the subject matter as a whole would have been obvious to a POSITA at the time of the invention. I have also been informed by Cisco's counsel that the obviousness analysis considers factual inquiries, including the level of ordinary skill in the art, the scope and content of the prior art, and the differences between the prior art and the claimed subject matter.

25. I have been further informed by Cisco's counsel that there are several recognized rationales for combining references or modifying a reference to show

obviousness. These rationales include: (a) combining prior art elements according to known methods to yield predictable results; (b) simple substitution of one known element for another to obtain predictable results; (c) use of a known technique to improve a similar device (method, or product) in the same way; (d) applying a known technique to a known device (method, or product) ready for improvement to yield predictable results; (e) choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success; and (f) some teaching, suggestion, or motivation in the prior art that would have led a POSITA to modify the prior art or to combine prior art teachings to arrive at the claimed invention.

V. BACKGROUND

A. TCP/IP and OSI Models

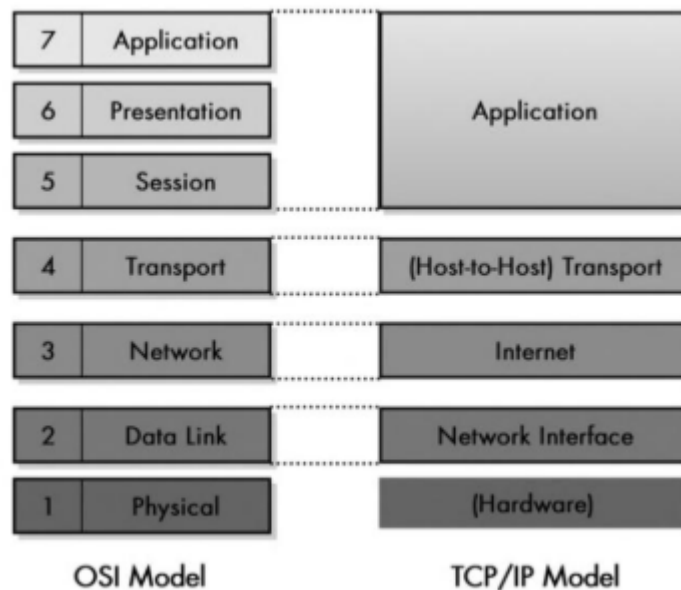
26. Computer networking is a complex task, one that is made easier by partitioning it “into smaller subtasks, traditionally by dividing the problem into layers.” Perlman, 1. The idea behind layering is that “each layer is responsible for providing a service to the layer above by using the services of the layer below.” Perlman, 1.

27. One of the most common sets of networking protocol layers is the Open Systems Interconnection (“OSI”) model. It was the original standard description for how packets should be transmitted between any two points. Tanenbaum, 28, 36. The

OSI model has seven layers: physical, data link, network, transport, session, presentation, and application. Kozierek, 129.

28. The physical layer defines electrical aspects of activating and maintaining physical links in networks. The physical layer represents cables, connectors, transceivers, and hardware. Kozierek, 102. The data link layer is where many wired and wireless LAN technologies function and specifies how packets access links and are attached to different frames when entering a new network environment. Kozierek, 103-04; Olderdissen, [0012]. The network layer specifies networking aspects, and handles the way that packets are addressed and forwarded. Olderdissen, [0012]. The transport layer handles the details of data transmission such as segmentation, packaging, reassembly, and flow control. Kozierek, 108. The session layer sets up and coordinates sessions between different machines and the applications at each end. Tanenbaum, 50. Another basic task of this layer is “dialog control.” Tanenbaum, 50. The presentation layer converts incoming and outgoing data from one presentation format to another for various reasons such as data compression, data encryption, and data translation. Tanenbaum, 51. This layer of the protocol facilitates the interoperability between the different encoding methods. Tanenbaum, 51. The application layer is the layer used to implement the functions performed by user applications and other high-level functions. Kozierek, 111-12.

29. The basic structure of communication networks is represented by the *Transmission Control Protocol/Internet Protocol (TCP/IP)* model. Tanenbaum, 28. This model is structured in five layers: hardware, network interface, internet, transport, and application. Kozierok, 129. The session, presentation, and application layers of the OSI model are condensed into the application layer of the TCP/IP model. Kozierok, 129. Other layers of the TCP/IP model track onto the OSI model as shown in Figure 8-2, Kozierok. The hardware layer of the TCP/IP model is the underlying network hardware. Kozierok, 128. The data link layer of the TCP/IP model is equivalent to the link layer and is used to interface with the network hardware. Kozierok, 128-29.



Kozierok, Fig. 8-2

30. TCP/IP has two primary protocols at the transport layer: TCP, which requires a connection between two devices, and UDP, which is connectionless. Kozierok, 17. Numerous protocols exist at the application layer including Hypertext Transfer 824Simple Network Management Protocol (SNMP), and Domain Name Service (DNS) for administrators. Kozierok, 130.

31. Each layer (irrespective of model) at a source node communicates with its peer layer at a destination node using a protocol. Perlman, 1. It is common for a single node to support several protocols and for a single layer to support several protocols as well. Perlman, 3.

32. In the connection-based TCP protocol, a connection is established between two devices that wish to communicate, vis-a-vis a communication session. Kozierok, 752. After the communication session has concluded, the connection between the two devices may also be terminated, e.g., closed. Kozierok, 761. In a standard scenario, the device that wishes to close the connection sends a packet to the other device that contains the FIN TCP header flag, and the other device responds by sending an ACK, followed by its own FIN message. Kozierok, 763, Fig. 47-5. *See also* Nayak, 752, 760-761, 763, Fig. 47-5; U.S. Patent Application No. 2006/0248582, [0095] (“[A] TCP endpoint that desires to initiate session closing operations sends a FIN command to the other endpoint. A FIN is represented by a TCP header flag.”); U.S. Patent Application No. 2004/0013112, [0109] (“A TCP

session is closed by both sides sending packets comprising the FIN flag indicating that the sender has no more data to send.”); RFC 793, 12 (“The clearing of a connection also involves the exchange of segments, in this case carrying the FIN control flag.”), 23 (Fig. 6 showing “CLOSE” actions accomplished by “s[e]nd FIN”), 38 (connection closing is initiated “by sending a FIN control signal”), 75 (if a packet received where “the FIN bit is set, signal the user ‘connection closing’”).

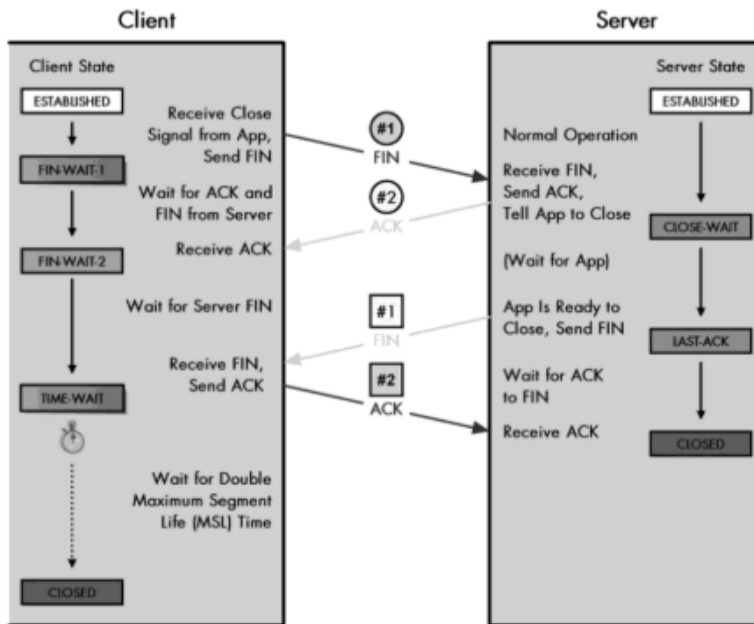


Fig. 47-5, Kozierok, 765

33. In a situation where one device wishes to end the connection, but the other device is unresponsive, the device attempting to close the connection may send

the other device a packet that contains the RST flag in the TCP header to force the connection to close. Kozierok, 760.

B. Packet Encapsulation

34. *In communications networks*, data is formatted differently according to the protocol utilized by the network, but most protocols have a relatively uniform structure. A given “message” that is sent across a network on links will generally consist of a header, the data, and the footer. Kozierok, 19, Fig. 1-3.



Figure 1-3, Kozierok

35. For example, a layer protocol may receive a packet and add a header to the information it received in a process called encapsulation. Perlman, 4. The header is like an envelope for a letter, with information such as a source and destination addresses, which help deliver the letter to its destination. Perlman, 4. As the packet is passed down layer by layer, protocols operating within those lower layers may add their own headers to the packet. Tanenbaum, 19 (“A message, *M*, is produced by an application process running in layer 5 and given to layer 4 for transmission. Layer 4 puts a header in front of the message to identify the message and passes the

result to layer 3. The header includes control information, such as sequence numbers, to allow layer 4 on the destination machine to deliver messages in the right order if the lower layers do not maintain sequence.”). Information contained in the packet header, such as a TCP port number or IP address is considered packet metadata. U.S. Patent Application No. 2017/0230065, [0030] (“traffic metadata, such as MAC address, IP address, TCP port, UDP port, etc.”); U.S. Patent Application No. 2014/0351106, [0033], (“a port number (e.g., a Transmission Control Protocol (TCP) port number) may also be included in the metadata record”); U.S. Patent Application No. 2012/0230208, [0057] (“metadata extracted from each outgoing TCP packet may include source and destination IP addresses, source and destination TCP ports”).

36. The actual data to be transmitted is called the “payload.” A packet’s “footer” is information that is placed after the payload and can be used in a similar fashion to the header in certain protocols. Kozierek, 19.

37. TCP is responsible for ensuring both that message data is divided into packets by the sending party, and that it is reassembled by the receiving party. U.S. Patent Application No. 2003/0014624, [0003]. TCP/IP is widely used for transmitting information as packets. U.S. Patent Application No. 2006/024858, [0015]; U.S. Patent No. 7,406,522, 1:59-60; Perlman, 4; Tanenbaum, 19; Kozierek, 19. A router or network device would contain a forwarding plane that would receive

information such as source and destination IP addresses and protocol type of incoming packets, and forward packets with minimal delay. U.S. Patent Application No. 2005/0281192, [0038] (“excessive delay” is a “forwarding plane failure”); U.S. Patent Application No. 2011/0110328, [0030] (“achieving lower latency” is an advantage of a forwarding plane); U.S. Patent Application No. 2013/0329578, [0029] (describing forwarding plane architectures “to reduce latency and other delays”).

C. Encryption

38. If information is transmitted in plain text, confidential or sensitive information could be read by someone other than the intended recipient, which might include an entity with a nefarious intent. Nayak, 163. Encryption is one method of disguising data to make it less likely to be read by a third-party using cryptography. Nayak, 163. At a relatively high level, encryption of data is performed using a “key” that is used to scramble the data, thereby rendering the data unreadable to those who do not have the key. Nayak, 164. The corresponding key that can decrypt encrypted data might be the same or different than the key used to encrypt data in the first place, depending on the specific encryption mechanism being employed. Nayak, 165. Typically, only the endpoints to the communication would typically have the decryption key. For example, the decryption key may be a “public key” which is known to both the sender and receiver, or a “private key” which is known to only

one party. Nayak, 167-69. Keys may also be used for authentication—a packet could be encrypted with a private key unique to the sender, so the packet’s origin can be verified. Nayak, 171. *See also* Nayak, 243-247; U.S. Patent Application No. 2011/0106710, [0067] (“As should be understood, because the PIN is encrypted by the terminal using a key that is not know[n] by the merchant or acquirer, the merchant or acquirer is unable to decrypt the PIN.”); U.S. Patent Application No. 2011/0022835, [0056] (“if an unauthorized client device 110 receives this message, it is unable to decrypt message 178 because it does not have possession of decryption key 112.”); U.S. Patent No. 6,944,476, 1:25-36.

39. Encryption can be performed at different layers in the protocol stack. Kozierek, 111. For example, the Secure Sockets Layer (SSL) protocol is usually associated with (and implemented within) the presentation layer. Kozierek, 111. Internet Protocol Security (IPsec) is a set of protocols that may similarly be implemented at lower layers in the protocol stack. Kozierek, 111. Encapsulating Security Payload (ESP) is another example of an encryption IPsec protocol that can be used to encrypt the payload of the packet, while the header and footer of the packet are left unencrypted. Kozierek, 466-71. Therefore, even without a key to decrypt the packet’s payload, a receiving or routing device could still authenticate a packet based on the header and footer information. Kozierek, 470.

D. Intrusion Detection System (IDS)

40. An IDS is used to detect and block malicious activity. Nayak, 225. Generally, an IDS will inspect both the packet header and the payload. Nayak, 225.

41. There are various detection mechanisms that an IDS may utilize, including signature-based detection, anomaly-based detection, or stateful protocol analysis. Nayak, 229. In most modern systems, an IDS is paired with an Intrusion Prevention System (IPS) in a combined system sometimes called an IDPS that identifies the attack (using the IDS functionality) and takes action to drop the packet, deny entry to the packet, or block the connection (using the IPS functionality). Nayak, 228. An IDPS analyzes network behavior and prevents attacks in real-time with active monitoring. Nayak, 229.

42. In signature-based detection, the IDS compares the traffic with known signatures for possible attacks, to determine if the incoming traffic matches a known attack signature. Nayak, 230. In anomaly-based detection, the IDS protects a system from unknown threats by establishing a “baseline” of normal traffic through studying the traffic over a given time period. Once the baseline is established, the IDS can compare the baseline traffic with detected traffic and identify abnormalities or deviations from the baseline as a possible threat. Nayak, 230. These anomalies may be anomalies in the protocol format and behavior, such as unusual TCP segmentation, incorrect IP fragmentation and reassembly, incorrect source and

destination port numbers, or flags in the TCP or IP header. Nayak, 232. Stateful protocol analysis involves creating predetermined profiles for permissible network activity that corresponds with protocols from standard bodies. Nayak, 233. An IDPS that performs stateful protocol analysis can track protocol states at both the network and application layers to track if a protocol is within the standards for acceptable protocol behavior. Nayak, 233.

43. When a packet's payload is encrypted, an IDS cannot base its analysis and determination on the payload, making it more difficult to effectively identify and block intrusion attempts. Nayak, 242. However, the IDS can still analyze flow information and the packet header. Nayak, 232-37.

44. One way to mitigate potential attacks based on the information available in a packet's header is by throttling a communication session by manipulating the TCP congestion window size. U.S. Patent Application No. 2002/0046264, [0079] ("by regulating the advertized window size of each system user, each user's bandwidth can be controlled"); U.S. Patent Application No. 2002/0059435, [0123] ("flow control can be applied... by shrinking the TCP windows being advertised"); U.S. Patent No. 6,826,620, 11:30-45 (an exemplary rate control technique where a "FRAU [frame relay access unit] controls... the TCP window size within the acknowledge packets.").

VI. OVERVIEW OF THE '824 PATENT

45. The '824 patent relates to management of private data during communication between a remote server and a user's device. '824 patent, Abstract. The '824 patent describes that the user's device receives a request from the remote server for retrieval of at least one data packet from the user's device. '824 patent, Abstract. The content of the data packets may be encrypted. '824 patent, 12:40. The user's device is "configured to provide a response corresponding to the received request." '824 patent, 3:3-4.

46. Characteristics of a data packet may be used to identify the packet's "data type" and the content of the data packet. '824 patent, 9:9-13. The characteristics of a data packet include:

[H]eader, footer, version number, request by hostname/IP, sent to hostname/IP, server hostname/IP, IP address, file extension, standard title, non-standard title, encryption yes/no, encryption method, encoding yes/no, encoding method (e.g., 'UTF8'), whole file, chunked file, keywords, API version, SDK version, Social plug-in version, driver, communication protocol, action verbs, HTTP/HTTPS, URL Part Names, and/or any other specific character T within the at least one data packet.

'824 patent, 8:66-9:9.

47. Similarly, at least one “dynamic feature” of a data packet may be used to identify the packet’s “data pattern.” ’824 patent, 9:60-10:5.

Dynamic features of a data pattern include “frequency (e.g., appearance(s) per second), size (e.g., in kilobytes), speed (in kilobytes per second), count (e.g., numbers), ratio (e.g., compared to other datatype flow), repeat (e.g., cyclic order), order, server reference (e.g., website reference such as “example.com/feature”) and/or any combination thereof and/or other pattern behavior ‘m’ within the at least one data packet.

’824 patent, 9:60-10:2.

48. The user’s device may have a “privacy preference,” which contains a list of allowed data packet types or allowed data packet patterns for sharing during communication. ’824 patent, 16:13-16.

49. Whether a data packet is shared depends on whether the data packet matches the data types or data patterns specified in the privacy preference. ’824 patent, 3:4-8, 3:3-6.

50. If the privacy preference dictates that a certain “data packet is forbidden for sharing” due to the data type and/or data pattern being incompatible with the privacy preferences, the content of the data packet can be modified. ’824 patent, 14:6-10. Alternatively, if the privacy preferences do not recognize the packet type, the packet can be blocked. ’824 patent, 13:4-11.

51. As I will explain below, these concepts were well known as of the priority date of the '824 patent.

VII. CLAIM CONSTRUCTION

52. It is my understanding that to properly evaluate the '824 patent, the terms of the claims must first be interpreted. I have been informed by Cisco's counsel that for the purposes of this *inter partes* review, the claims are to be construed under the so-called *Phillips* standard, under which claim terms are given their ordinary and customary meaning as would have been understood by a POSITA in light of the specification and prosecution history, unless the inventor has set forth a special meaning for a term. I have also been informed that claim terms only need to be construed to the extent necessary to resolve the obviousness inquiry.

53. I have reviewed the entirety of the '824 patent, as well as its prosecution history. It is my opinion that for purposes of applying the asserted prior art to evaluate the patentability of the Challenged Claims, no terms require express construction.

VIII. IDENTIFICATION OF HOW THE CLAIMS ARE UNPATENTABLE

54. The discussion in this Declaration provides a detailed analysis of how the asserted prior art references teach each limitation of the Challenged Claims.

55. As part of my analysis, I have considered, and discuss in detail, the scope and content of the prior art and any differences between the alleged invention and the prior art.

56. It is my opinion that the alleged invention recited in the Challenged Claims would have been obvious in view of the teachings of the asserted prior art and the knowledge of a POSITA.

IX. THE COMBINATION OF BURNS AND YANG RENDERS OBVIOUS CLAIMS 17-20

A. Summary of Burns

57. Burns is directed to identifying whether an encrypted packet is associated with a network attack, and either responding to the attack by, for example, blocking the packet, or forwarding the packet. Burns, Abstract. Burns' intrusion detection system (IDS) identifies if an application is associated with a packet, determines if the packet is encrypted, and determines if a packet is associated with a network attack. Burns, Abstract, 2:51-3:33. If a packet is not associated with a network attack, IDS may forward the packet. Burns, Abstract. If a packet is associated with a network attack, the IDS takes action and can either drop the packet, terminate the session, or block future sessions from the computing device. Burns, 5:61-63.

58. To determine if a packet is encrypted, Burns's IDS can employ multiple techniques via an encryption detection module. Burns, 9:8. The encryption detection

module may extract the application-layer header and the application-layer payload from the packet. Burns, 12:10-16. The encryption detection module analyzes the packet header and the payload to determine if the randomness within the data indicates that encryption is present.¹ Burns, 12:20-25. Burns describes multiple methods for determining the randomness of a packet via the Wald-Wolfowitz runs test algorithm, or through additional tests and filters that can be added, removed, or edited. Burns, 9:54-59.

59. Burns describes that potentially malicious applications may encrypt all or portions of an application-layer header and/or application-layer data (payload). Burns calls applications where both the header and payload are encrypted “fully encrypted.” Burns, 5:1-4. When fully encrypted packets are detected, Burns describes that the packets can be dropped, the traffic may be throttled, details about the communication session can be logged, or an alert message can be sent. Burns, 6:49-56.

60. If a packet is fully encrypted, the IDS can also determine if the packet includes key exchange information. Burns, 13:40-42. If key exchange information is provided, the IDS can determine that the packet, although it may be fully

¹Randomness strengthens cryptographic encryption—if an encryption algorithm has high randomness, it becomes more difficult for a third-party to “break” the encryption and decipher the encrypted data. *See* Nayak, 172-180.

encrypted, is not associated with an invalid or unwanted application. Burns, 13:43-46.

61. In contrast, legitimate applications following well-known protocols generally do not encrypt the application-layer header, the TCP header, or the IP header. Burns, 5:7-13. If a packet is not fully encrypted, Burns describes forwarding these packets or sending these packets to an attack detection module. Burns, 11:25-29.

62. Burns and the '824 patent are both directed towards dynamic management of private, or encrypted, network communications. '824 patent, Claim 1, 2:64-67; Burns, 2:47-49, 5:53-6:8, 6:46-56. Both Burns and the '824 patent describe a method by which a user or administrator may selectively block (or modify) packets that do not correspond with their settings. *See* '824 patent, 2:27-28 (“wherein the privacy preference includes a list of allowed data patterns for sharing during communication”), 2:12-15 (“wherein the privacy preference includes a list of allowed data packet types for sharing during communication”), 12:8-10 (“may perform at least one of sharing, blocking and/or modifying... according to the corresponding privacy preference”); Burns, 5:30-34 (“a system administrator may configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a specified list of identifiable applications, or prevent all communications...”), 6:50-51 (“the user interface may permit the administrator to

drop all fully encrypted packets...drop the packets only if a key exchange has not been previously identified for the communication session...”). Burns is also directed to addressing the same problem of controlling the transmission of a user’s private data by restricting the use of encrypted communications. Burns, 1:21-35, 2:21-26.

B. Summary of Yang

63. Yang, like Burns, describes an IDS that can determine whether packet flow represents a network attack. Yang, Abstract. The IDS consists of a flow analysis module, a stateful inspection engine, and an application identification module. Yang, 6:29-36.

64. The flow analysis module receives inbound traffic and uses source media access control (MAC) address, destination MAC address, source port, destination port, and IP addresses to identify network flows. Yang, 6:60-7:2.

65. The stateful inspection engine buffers a copy of the packet flow and reassembles it to determine whether the packet flow is associated with an identifiable application. Yang, 7:16-20. The stateful inspection engine can determine if a packet is encrypted. Yang, 7:27-28. If a packet is encrypted, it can determine whether the packet flow is associated with an unwanted application, and whether key exchange has occurred. Yang, 7:23-32. If an unwanted application is detected, the stateful inspection engine can block a communication, output an alert, or take on different

actions. Yang, 7:45-54. If no security risk is detected, the stateful inspection engine directs the forwarding component to forward the packet flows. Yang, 7:51-54.

66. Once a communication session has been interrupted by the stateful inspection engine, the application identification module can attempt to identify an application associated with an intercepted session. Yang, 8:11-13. The application identification module sends data from the communication session to a protocol decoder, which utilizes an application-layer header of a packet to identify a protocol being used by the communication session. Yang, 8:24-44.

67. Yang, like Burns and the '824 patent, is directed towards dynamic management of private, or encrypted, network communications. '824 patent, Claim 1, 2:2-3, 4:66-67; Yang, 2:47-49, 5:54-6:8, 6:46-56. Yang and the '824 patent describe a method by which a user or administrator may selectively forward or block packets that do not correspond with the administrator's settings. 2:27-28 ("wherein the privacy preference includes a list of allowed data patterns for sharing during communication"), 2:12-15 ("wherein the privacy preference includes a list of allowed data packet types for sharing during communication"), 12:8-10 ("may perform at least one of sharing, blocking and/or modifying... according to the corresponding privacy preference"); Yang, 5:53-65 ("administrator may specify a compound network attack... IDS 10 may take one or more programmed actions, such as automatically dropping packet flows"). Yang is also directed to addressing

the same problem of controlling the transmission of private data, such as by restricting communications to and from a private computing devices in a private enterprise computing network. '824 patent, 3:11-13; 4:4-6; Yang, 4:17-45; 5:53-65.

C. Reasons to Combine Burns and Yang

68. A POSITA would have combined Burns and Yang because Burns refers to and incorporates Yang by reference. Burns, 4:39-45.

Exemplary techniques for identifying specific applications and protocols are described in greater detail in U.S. patent application Ser. No. 11/835,923, Burns et al., "Identifying Applications for Intrusion Detection Systems," filed Aug. 8, 2007, assigned to the assignee of the present application, which is hereby incorporated by reference in its entirety.

Burns, 4:39-45.

69. The Yang patent (8,291,495) ultimately issued from the 11/835,923 application that was referred to in Burns. Burns therefore encourages the combination of its disclosure with Yang, and states that Yang provides "exemplary techniques for identifying specific applications and protocols." Burns, 4:39-45. Burns describes an IDS that is responsible for "monitor[ing] traffic into and out of an enterprise network." Burns, 2:52-54. Yang's disclosure is directed to "techniques for detecting and preventing network attacks ... network viruses or other malicious activity. Yang, 1:45-49. Yang adds crucial implementation details that Burns expects

but does not expressly address. In fact, implementing Burns by itself would result in an IDS that can identify a threat, but does not have capability to address it. A POSITA would have seen this to be a shortcoming and would have looked to a reference like Yang to improve Burns' IDS to provide it with the functionality to respond to a threat, not just identify it. To ensure that an IDS would detect and prevent attacks or other malicious activity, it would have been obvious to incorporate the disclosure of Yang into the disclosure of Burns.

1. Claim 17

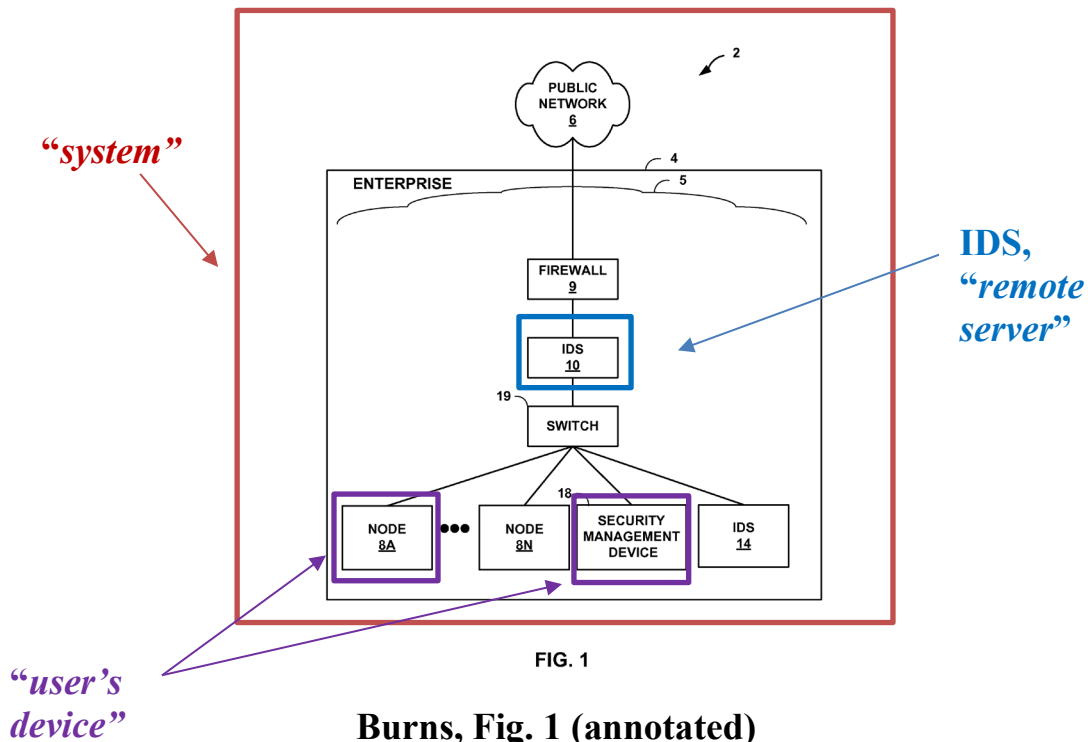
- a. **[17.0] *A system for dynamic management of private data during communication between a remote server and at least one user's device, the system comprising:***

70. Burns discloses the dynamic management of private data, such as encrypted data. A POSITA would have recognized that encrypting a packet would have prevented a third party who did not have the encryption key from reading it. Nayak, 163-67. Burns, in its disclosure of encrypted data, renders obvious "*private data.*" This interpretation is supported by the Ferreira reference cited in the '824 patent, which states:

The article presents a solution for storage and **management of private data**... The approach is based on a middleware architecture supported by homomorphic **encryption techniques** combined with dynamic indexing mechanisms.

Ferreira, Abstract.

71. Burns describes “blocking unidentified encrypted communication sessions.” Burns, Abstract; *see also* Burns, 9:60-63, Fig. 4 (“flowchart illustrating an exemplary operation of an IDS [intrusion detection system]”). In Fig. 1, Burns illustrates a “system” that includes an “intrusion detection system,” or “IDS.” The IDS “attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices in public network 6.” Burns, 4:36-39.



Burns, Fig. 1 (annotated)

72. Part of the system includes “internal computing nodes” 8 which represent any “private computing device within enterprise network 5, including

workstations.” Burns, 4:32-34. It would have been obvious for a computer in an enterprise network, such as a workstation, to be used by a user. A given node (e.g., node 8A), computing device or workstation corresponds to the claimed “a user’s device.” Alternatively, Burns describes configuring the IDS via security management device 18. Burns, 6:11-13. It would have been obvious for security management device 18 to also be a workstation computer similar to node 8A that is used by an administrative user. See Burns, 6:5-8, 6:37-38. Thus, security management device 18 also corresponds to “a user’s device.”

73. The IDS of Fig. 1 is shown to be separated from the node 8A (*user’s device*) and security management device 18 by the switch 19. Burns explains that the IDS may be implemented within a “network device,” thereby making it a *server*. Because the IDS and node are separate, a POSITA would have understood the IDS to correspond to a “*remote server*.” See, e.g., Nayak, 238-42 (showing different configurations of a switch separating an IDS from devices.)

74. Burns further explains that its IDS “attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices.” Burns, 4:37-38. Therefore, a POSITA would have appreciated that Burns teaches “*communication*” between the IDS (*remote server*) and the nodes (*user’s device*). Alternatively, since Burns contemplates the IDS presenting a user interface to an administrative user at security management device 18 (Burns, 6:5-8,

6:11-13, 6:37-38), it would have been obvious for there to be “*communication between*” security management device 18 and the IDS.

b. [17.1] a memory;

75. Burns’s system describes different types of memory, including: “random access memory (RAM),” “a hard drive, a network drive, [or] a flash memory stick.” Burns, 8:4-6, 18:51-53; *see also* 21:24-31.

c. [17.2] a communication data type database, comprising at least one communication data type corresponding to sharing of at least one data packet from the user’s device;

76. Burns describes that its IDS “attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices in public network 6.” Burns, 4:36-39. This is performed by “stateful inspection engine 28 of forwarding plane 22,” which is part of the IDS (“*the remote server*”). Burns, 6:29-36, Fig.2. Burns references Yang’s disclosure for greater detail regarding “[e]xemplary techniques for identifying specific applications and protocols.” Burns, 4:36-42.

77. Yang describes identifying the “type of application and protocol associated with the packet flow” through static port mapping with an “application identification module.” Yang, 9:46-10:29. Yang’s application identification module uses a packet’s TCP port number to identify certain applications and/or protocols. Yang, 9:59-10:24. Yang provides numerous examples of a TCP port number that

correspond to a certain application, or *data type*. Yang, Table 1. A POSITA would have understood this table to correspond to a “*communication data type database*.”

TABLE I

PORT	APPLICATION
20	FTP
22	SSH
23	Telnet
25	SMTP
43	WHOIS
53	DNS
67	BOOTP or DHCP
70	Gopher
79	Finger
80	HTTP
109	POP
110	POP3
113	ident/IRC
118	SQL
119	NNTP
194	IRC
443	HTTPS
445	SMB
564	RTSP

Table 1, Yang

78. POSITA would have known that Yang’s “static port mapping” approach allows for an association between the port number (a characteristic) and certain types of “content,” such as an application or protocol. For example, for the port value 80, HTTP corresponds to “Hypertext Transfer Protocol (World Wide Web)” and indicates that a packet contains website data. Kozierok, 709. For 443, HTTPS is the encrypted and therefore “secure” HTTP. For 109, POP corresponds to “Post Office Protocol” and indicate that a packet contains email data. Kozierok, 709. See also Kozierok, Table 43-1 (showing well-known port numbers and their

corresponding applications); and RFC 1700, (listing port numbers for thousands of known applications.)

79. A POSITA would have been familiar with the format of TCP and IP packet headers and would have recognized that identifying an application or protocol with Yang's technique would use multiple packet header values (characteristics). U.S. Patent No. 7406522, 1:59-60; U.S. Patent Publication No. 2006/0209858, [0015]; U.S. Patent Publication No. 200/30014624, [0003]; Perlman, 4; Tanenbaum, 19; Kozierok, 19. First, Yang's technique requires identifying that an IP packet contains a TCP packet via the IP packet header's "protocol" field. Yang, 5:16-18, 6:24-27; see RFC791, 14 ("Protocol:... This field indicates the next level protocol used in the data portion of the internet datagram."); RFC790, 6 (showing TCP assigned Internet Protocol Number 6). Second, Yang's technique identifies an application or protocol via the source and destination port numbers in the TCP packer's header. Yang, 6:54-58, 17:3-5; see RFC793, 15 (showing "Source Port" and "Destination Port" in the "TCP Header Format").

80. In the combination, Burns's IDS (*remote server*) would receive a packet. Yang's application identification module would use static port mapping to identify a port. The port number would correspond to an application, which would be used by Burns's IDS to determine how the packet would be routed in subsequent steps.

81. It would have been obvious to a POSITA that the data types in the static port mapping approach would correspond to “*sharing of at least one data packet from the user’s device,*” because TCP is a bidirectional protocol. Kozierok, 721.

d. [17.3] *a privacy preference database,*

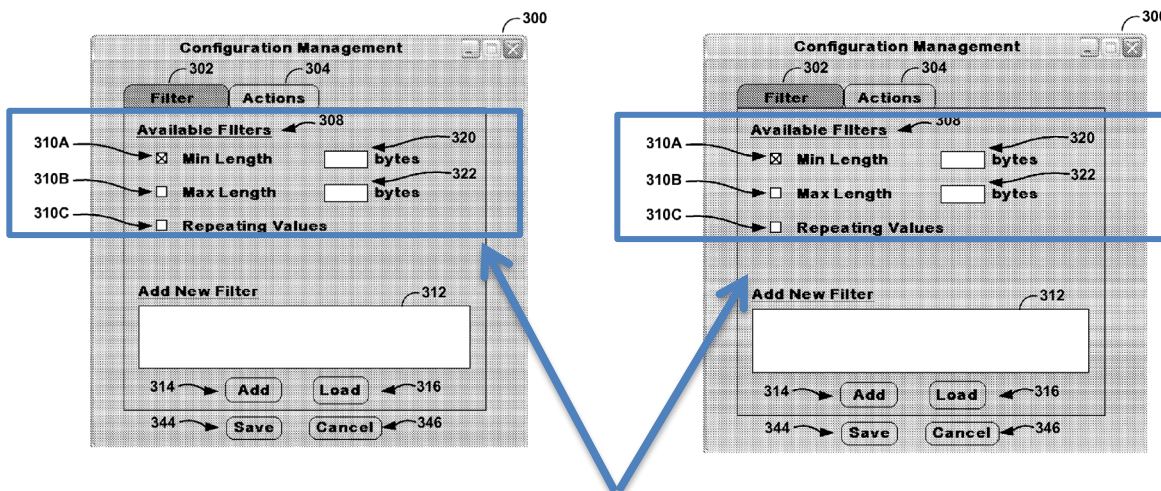
82. Burns’s IDS can be tailored to an administrator’s desired configuration information. Burns, 1:21-35. Burns allows a system administrator to “configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a specified list of identifiable applications, or prevent all communications.” Burns, 5:30-34. Furthermore, an administrator can specify attack definitions and desired responses:

Initially, security management module 44 receives configuration information from administrator 42 and, in response, configures IDS 20 to monitor a network or portions thereof (subnets) of interest... configuration manager 44 presents a user interface by which **administrator 42 specifies patterns or other attack definitions 33.** For example, administrator 42 **may configure IDS 20 to block all packets deemed to be fully encrypted and to log information about the communication session associated with the blocked packets.** Administrator 42 **may also configure IDS 20 to throttle down (i.e., bandwidth limit) the communication session** associated with the packets to minimize the bandwidth used by the communication session. Burns, 9:64-109.

Security management module 33 **may create this log in, e.g. a database,** a text file, or any appropriate data structure.

Burns, 20:65-67.

83. The administrator's configuration information allows the administrator to explicitly permit certain applications, deny others, and provide an "attack definition." A POSITA would have recognized that, for example, "block[ing] all packets deemed to be fully encrypted" would be a "privacy preference" as it prevents encrypted, (*private*) and potentially malicious packets from reaching the user's device.



Configurable privacy preference items

Burns, Figs. 7A-7B (annotated)

84. A POSITA would have found it obvious to store this information in a "database" because a database is a common location to store desired configuration information.

- e. **[17.4] comprising a list of allowed types of data packets for sharing during communication with the at least one user's device;**

85. As discussed in the analysis of [17.2], a packet's identified application corresponds to its "*data type*." The IDS 10 can be configured to allow or block packets that are associated with specific applications:

Where IDS 10 is able to identify the application using a particular communication session, IDS 10 may **either permit or prevent** the communication session from continuing. For example, a system administrator may configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a **specified list of identifiable applications**, or prevent all communications except for communications from a **specified list of permitted software applications**.

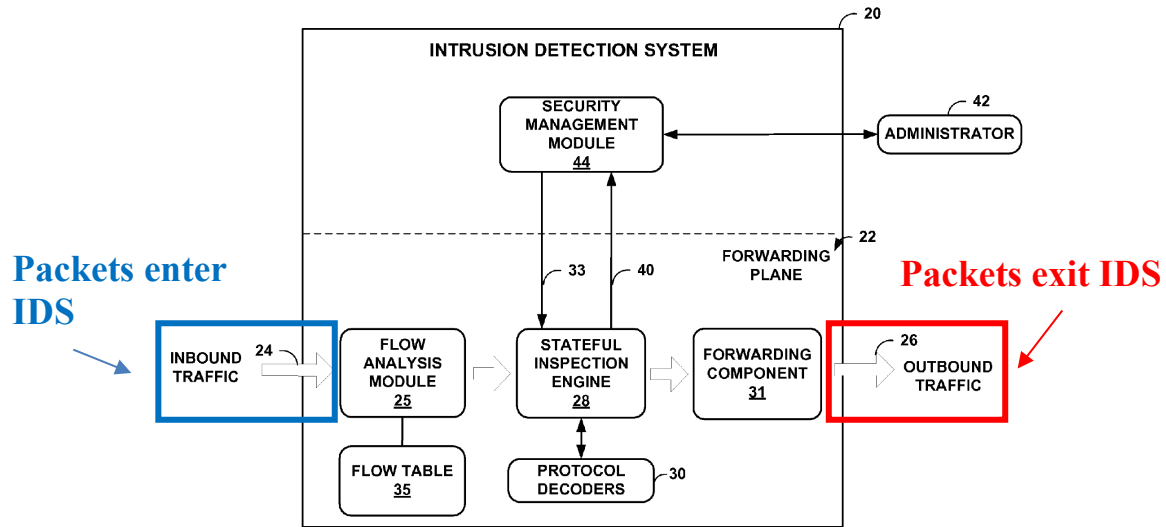
Burns, 5:27-34.

86. The administrator's configuration information that contains a "specified list of identifiable applications" to permit or prevent renders obvious the claimed "*list of allowed types of data packets for sharing during communication with the at least one user's device*."

- f. **[17.5] a communication module, to allow communication between the remote server and the at least one user's device; and**

87. In Fig. 2, Burns shows that network traffic flows into and out of the IDS, wherein "each network flow represents a flow of packets." Burns, 6:61-63. The

IDS contains elements that would be part of a forwarding plane, such as the “forwarding component 31”, thus, the IDS corresponds to the claimed “communication module.” Burns, Fig. 1.



Burns, Fig. 2 (annotated)

88. As discussed at [17.0], Burns shows in Fig.1 that the IDS manages communications “between computing nodes 8 and other computing devices in public network 6.” Burns, 4:36-39. Thus, it would have been obvious for the IDS’s network interface(s) to “allow communication between the remote server [i.e., the IDS itself] and the at least one user’s device.”

g. [17.6] *a processor, coupled to a response database and to the privacy preference database,*

89. Burns describes a “programmable processor” that receives a network packet. Burns, 3:37. Burns further describes that the IDS may be implemented using a “general-purpose processor”:

Methods described herein may be performed in hardware, software, or any combination thereof within a network device. For example, methods described herein may be performed by an application specific integrated circuit (ASIC) or a general-purpose processor. Methods described herein may also be embodied in a computer readable medium containing instructions. Instructions embedded in a computer readable medium may cause a programmable processor, or other processor, to perform the method, e.g. when the instructions are executed.

Burns, 21:14-23.

90. The IDS operates in accordance with the administrator's configuration, as addressed at [17.3]. It would have been obvious for Burns's programmable or general-purpose processor to have access to the administrator-provided configuration, so the administrator's configurations would instruct the IDS.

91. The claimed "*response database*" lacks antecedent basis. However, two possible interpretations for this database may exist. First, Burns describes that the IDS contains a user interface for the administrator to select "response action[s]." Burns, 19:61-66, Fig. 7B. Alternatively, Burns's IDS also contains executable code for performing response actions, which could also be considered the "*response database*." Burns, 3:17

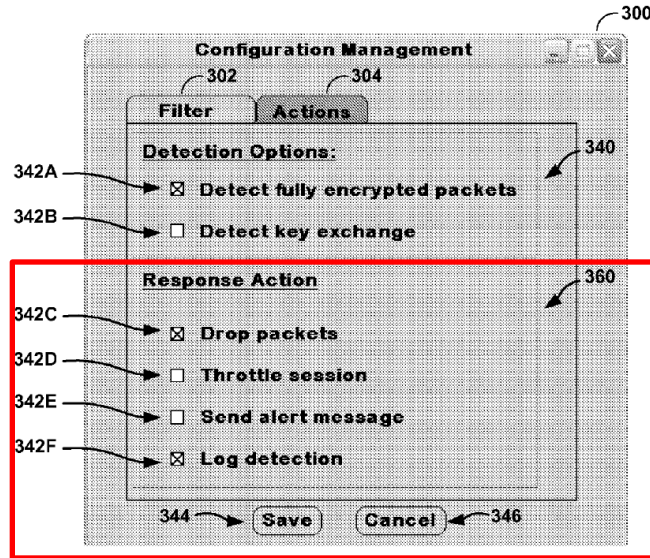


FIG. 7B

Burns, Fig. 7B, annotated

[17.7] wherein the processor is configured to instruct the remote server

92. As addressed above at [17.6], Burns’s system involves a processor configured to work with the IDS. As addressed at [17.0], Burns’s IDS corresponds with the “remote server.” It would have been obvious for the processor to instruct the IDS, because a processor is the “brain” of a computer and is responsible for directing its operations. *See, e.g.*, U.S. Patent Publication No. 2003/0038791, [0025] (“central processing unit 22 is a brain of computer system” that will “direct the operation of the internal processor components.”); U.S. Patent No. 5289589, 1:17 (“Processors are the brain of computer systems...”); U.S. Patent Publication No. 2006/0040650, [0031] (“Executed by a device’s processor(s), these instructions

direct the operation of the device. These instructions can be stored in a memory of the computer. Instructions can invoke other instructions.”). Burns discloses that “[i]nstructions embedded in a computer readable medium may cause a programmable processor, or other processor to perform the method,” wherein the method is the identification of and response to an encrypted packet. Burns, 21:20-22.

93. It would have been obvious to a POSITA that the “*processor*” would be a component within the “*remote server*,” an interpretation that is consistent with the ’824 specification. *See* ’824 patent, 13:36-37 (“In some embodiments, processor 402 may be embedded in server 401.”) Thus, Burns renders obvious the claimed “*processor*.”

h. [17.8] *to determine at least one data type for sharing of data packet that is compatible with the list of allowed patterns of data packets for sharing,*

94. There is no antecedent basis for the claimed “*list of allowed patterns of data packets for sharing*.” The limitation “*determine at least one data type for sharing of data packet that is compatible with the list of allowed patterns of data packets*” could correspond to two possible interpretations. (1) Allowed data types only: perhaps the claim was meant to refer to a “*list of allowed data types*.” (2) Data types and data patterns: the claim may have been intended to introduce a second, new, list of “*allowed data patterns*” in addition to determining “*at least one data*

type.” Per this interpretation, the first step would be to “*determine at least one data type for sharing of data packet*” that is compatible. The second step would be to determine if a “*data pattern*” of the packet is compatible with the “*allowed patterns of data packets for sharing.*”

95. (1) Allowed data types only: As discussed at [17.3], Burns and Yang describe how an IDS can determine the “*data type*” of a packet based on the packet’s “*characteristics.*” Burns describes that the IDS can analyze packets to determine an application or protocol that corresponds to a given communication session:

In one embodiment, IDS 10 attempts to identify **applications and protocols for each communication session** between computing nodes 8 and other computing devices in public network 6.

Burns, 4:36-39.

Application identification module 51 attempts to identify an application associated with the received packet (76). For example, application identification module 51 may inspect the packet header to determine the destination port of the packet. In some cases, **the destination port is associated with the protocol or application** being used for the communication session.

Burns, 10:32-38.

[A]pplication identification module 51 may use the well-known static port binding as a default application

selection. Table 1, below, shows an example static port binding list.

Yang, 9:64-66.

In order to identify network communications associated with unwanted applications, IDSs or other security systems may analyze packet streams of the network communications and employ behavioral analysis. For example, a security system may perform deep packet inspection and apply patterns to the pay loads of the packets in an attempt to identify **the source software application**.

Burns,1:36-42.

96. This is further detailed in Yang's method of identifying applications based on the TCP port of the packet's header. Yang, 9:46-10:24. A port number corresponds to a "*characteristic[] of the at least one data packet.*" The application that is identified from the port number corresponds to the "*data type.*" Yang describes that a packet's corresponding application may be "unknown," which is one "*data type*" identified by the '824 patent. Yang, 8:14-18; '824 patent, 9:23 ("unknown data type content"). Therefore, a POSITA would have recognized that the type of application in a packet is a "*data type.*"

97. Alternatively, the IDS determines whether a packet's contents correspond to a data type of "encrypted" or "plain text." It would have been obvious to a POSITA that whether a packet is encrypted corresponds to "*at least one data type.*" It was common in the art to refer to generically refer to a packet's

data type as being “encrypted.” *See* U.S. Patent Publication No. 2008/0159146, [0024] (referring to “encrypted” as a “high entropy data type[]”); U.S. Patent Publication No. 2007/0088845, [0010] (“data types like... encrypted data”); U.S. Patent Publication No. 2005/0047449, [0031] (referring to “encrypted multimedia data” as a “data type[]”).

98. As addressed at [17.3], Burns’s administrator configuration information corresponds to the “*privacy preference*.” Since the administrator’s configuration information is a list of which applications (*data types*) are allowed or prohibited, Burns and Yang render obvious the first possible interpretation of [17.8]: “*determine at least one data type for sharing of data packet that is compatible with the list of allowed data types of data packets.*”

99. **(2) Data types and data patterns:** the second possible interpretation of [17.8] involves both determining a “*data type for sharing*” and “*a list of **allowed patterns** of data packets for sharing.*”

100. Burns illustrates in Fig. 5 that the IDS may be configured to make determinations based on multiple factors, including whether the application or protocol is known (step 126) and whether a packet is encrypted (step 128). As discussed at [17.6], Burns’s IDS is configurable to act in accordance with an administrator’s preferences. Thus, it would have been obvious for the configuration information to include preferences relating to both applications and

protocols (“*data packet types*”) and whether a packet contains repeated values indicative of not being encrypted (“*communication data patterns*”). *See, e.g.*, Burns, 5:29-53, 14:32-38.

101. Burns illustrates in Fig. 5 that the IDS may be configured to make determinations based on multiple factors, including whether the application or protocol is known (step 126) and whether a key exchange is observed (step 130). As discussed at [17.4], Burns’s IDS is configurable to act in accordance with an administrator’s preferences. Thus, it would have been obvious for the configuration information to include preferences relating to both applications and protocols (*data packet types*) and whether a key exchange is observed (*communication data patterns*). *See, e.g.*, Burns, 5:29-53.

102. The prior art describes three examples of allowed “*pattern*”: (a) size, (b) encryption key exchanges, (c) repeated sequences or values.

103. (a) Size: Burns describes that a packet’s size may be used to determine how to classify a packet. Burns, 2:34-35 (“If the average size and frequency of data transmission matches known characteristics for a malicious or unwanted application, the IDS may block further communication of that session,”), 14:17-19 (“Another exemplary method for identifying encrypted packets includes tracking two classes for bytes of a packet and comparing the sizes of the classes,”), 14:42-50 (“Encryption detection module 58 may then determine the size N of the

TCP/IP payload portion or, in another embodiment, just a portion of the TCP/IP payload of the received packet that typically corresponds to an application-layer header (142). In general, the size of the packet's TCP/IP payload refers to the number of bytes in the packet to be analyzed. In one embodiment, encryption detection module 58 may receive the number of bytes in the packet as an argument to a function or as a configurable input from a user.”) Size and frequency are “*pattern[s]*.” See ’824 patent, 9:6410:6.

104. Burns describes that the IDS allows non-encrypted packets to pass through. Burns, 11:19-25. Therefore, if a packet of a certain size and frequency corresponded to a non-encrypted packet, a POSITA would have recognized that these packets would be allowed to pass through.

105. Similarly, Burns describes that a packet’s size could be a “configurable input” by the user. Because the user is able to configure the preferences to prevent certain applications or protocols from being forwarded, it would have been obvious that a user could also tailor their input to “*allow*” certain packets based on their size.

106. (b) Encryption key exchanges: Burns describes that whether an encryption key has been exchanged can be used to determine the “*pattern*” of a packet:

In particular, when "Detect key exchange" check box 342A is selected, administrator user interface 300 sends a message to security management module 44 to detect key exchanges for monitored communication sessions, or to operate in a detect key exchange mode. Accordingly, security management module 44 instructs stateful inspection engine 28 to **analyze incoming packets to determine whether those packets represent a key exchange** for their corresponding communication session. Stateful inspection engine 28 then begins to determine whether incoming packets represent a key exchange as part of the communication session. In one embodiment, stateful inspection engine 28 may only determine whether a first set of packets of a communication session represents a key exchange.

Burns, 19:25-38; *see also* 12:47-13:8.

107. A POSITA would have recognized that different protocols utilize different key exchanges and have associated patterns. For example, the SSH key exchange involves SSH_MSG_KEXINIT and SSH_MSG_NEWKEYS. RFC 4253 17 & 21. Burns uses SSL as an example of an encryption key exchange:

Protocol decoders 30 may further inspect the packet to determine if the packet is part of a key exchange of the communication session....As an example, the communication session may be in accordance with the secure socket layer (SSL) protocol. Although application identification module 51 may not be able to determine the specific application

associated with the communication, protocol decoders 30 may determine that the communication session is using the SSL protocol.... protocol decoders 30 may recognize a packet that includes key exchange information that is exchanged as part of the SSL protocol.

Burns, 12:42-13:2; *see also* 12:52-13:8 (describing SSL key exchange that requires a specific order of packets sent with ClientHello, ClientMasterKey, and ServerVerify messages to exchange keys).

108. Burns and Yang describe that one “*data type*” is categorizing a packet as an “unknown” application or protocol. Burns, 8:66-67, 18:43; Yang, 8:14-18. This is consistent with the ’824 patent, which refers to “unknown data type content.” ’824 patent, 9:13-17, 9-25-30. Burns describes that a communication session can be forwarded even for unknown applications if a key exchange has occurred:

In one embodiment, where IDS 10 determines that a communication session, for which an **application cannot be identified**, is encrypted, IDS 10 may further determine whether a key exchange associated with the communication session can be identified; **where a key exchange has been identified, IDS 10 may permit the communication session** and where a key exchange has not been identified, IDS 10 may terminate the communication session.

Burns, 5:64-6:4.

109. Burns and Yang reference other well-known protocols that would utilize encrypted packets. For example, the TCP port number approach in Yang

describes both SSH and HTTPS as potential protocols. Yang, Table 1. A POSITA would have recognized that other protocols could be determined from their port value. *See* Kozierok, 709.

110. It would have been obvious that even if a packet were sent using a protocol that was known to be encrypted, that the key exchange would allow this packet to be forwarded in the system of Burns or Yang. *See* Burns 13:37:41 (“protocol decoders 30 may have determined that one or more previously exchanged packet represented a key exchange and logged the fact that a non-hidden (i.e., readily detectable) key exchange has occurred for the communication session.”) Therefore, Burns and Yang render obvious a determination of the packet’s *data type* (a known encrypted application) in addition to a *data pattern* (key exchange) to determine how the packet should be treated.

111. (c): Repeated sequences or values: In Burns, a repeated sequence or value is a *pattern* that may be used to indicate that the packet is not encrypted and is therefore an “allowed” packet per the administrator’s configuration.

Another exemplary method for determining whether a packet is encrypted includes identifying repeating patterns of varying lengths, e.g., one byte, two bytes, four bytes, or other within a packet.

...

In one embodiment, encryption detection module 58 **compares data in the packet to other data in the packet to find a repeating sequence**. In one embodiment, encryption detection module 58 stores a sequence of data from each packet in a communication session and determines whether there exists **a repeated value in each packet of the communication session**; in this case, the repeated value may indicate a standard header of an unknown protocol.

Burns, 14:32-35, 18:30-44.

112. Yang provides a list of well-known applications and protocols that are known to not be encrypted and would be forwarded. Yang, Table 1, 10:5-24 (listing FTP, Telnet, Gopher, and HTTP). Because protocols and applications follow common formats, it would have been obvious that these certain protocols have repeated sequences and would correspond to an “*allowed pattern*.”

113. Burns describes that patterns table 54, attack definitions 33, and anomalies table 56 are used by the stateful inspection engine of the IDS. Burns, 7:62-64. Yang provides further details about how anomaly and pattern information may be used by the IDS to detect *data patterns*: “stateful inspection engine 28 combines pattern matching with protocol-specific anomaly analysis to detect sophisticated attack behaviors.” Yang, 11:49-51.

114. Therefore, Burns and Yang describe determining an application or protocol of a packet, (“*determine at least one data type for sharing of data packet*”) and detecting a repeating sequence to determine that a packet is not encrypted (“*compatible with the list of allowed patterns of data packets for sharing*”).

- i. [17.9] *wherein the at least one data type is determined in accordance with characteristics of the communication data packet, and*

115. As discussed at [17.2] and [17.8], the TCP port mapping approach described in Yang involves determining a port number, and using this port number to identify the application or protocol of the packet. Yang’s technique involves identifying that an IP packet contains a TCP packet based on the protocol field of the packet’s header. Yang, 5:16-18, 6:24-27; RFC 791, 14 (“Protocol:... This field indicates the next level protocol used in the data portion of the internet datagram.”); RFC 790, 6 (showing TCP assigned Internet Protocol Number 6). Yang’s technique further describes identifying the application or protocol based on source and destination port numbers in a packet’s header. Yang, 6:54-58, 17:3-5; RFC 793 (showing “Source Port” and “Destination Port” in the “TCP Header Format”). As addressed at [17.8], the application or protocol corresponds to a “*data type*.” The port number therefore corresponds to a “*characteristic[] of the communication data packet*,” rendering obvious this limitation.

- j. [17.10] *wherein the content of the at least one data packet is not read by the remote server for continued operation by the user's device in real time during communication between the remote server and the user's device.*

116. As explained in the analysis of [17.0], Burns's IDS (*remote server*) is in communication (sending and receiving a *data packet*) with node 8A (*user's device*). As addressed in the analysis of [17.1], it would have been obvious that the claimed "*private data*" would refer to encrypted data. Burns explains that packets received by its IDS may be encrypted. Burns 5:17-20.

117. It was common knowledge that encrypted "*content*", including within the packet payload, could not be read without an encryption key. Nayak, 163-167. A POSITA would have also known that an IDS would generally not have access to the encryption key needed to decrypt packets it receives and could therefore not read the encrypted content. *See, e.g.*, Nayak, 243-247. As Burns explains, packets are "encrypted... by one party and decrypted by the other party to a communication session." Burns, 2:1-3. Since Burns's IDS is not a "party" to the communication session itself, it would have been obvious to a POSITA for the IDS not to decrypt packets passing through it. Therefore, Burns's IDS (*remote server*) would not "*read*" the "*content of the data packet.*" This is in line with Yang's method of identifying an application or protocol based on TCP and IP values as described in [17.2] and [17.8], as these values would not have been encrypted. *See* Nayak, 232-37.

118. Burns's IDS is an example of an IDPS, as upon identification, it can "drop[] the packet" or "block[] future communication sessions." Burns, 5:61-63. It would have been obvious to a POSITA that the IDS would perform the functions of identifying and preventing potential attacks in *real-time*. This is consistent with a POSITA's knowledge that most modern IDSs have IDPS functionality that allows them to not only identify the attack, but also to take action to drop the packet, deny entry, or block the connection in real-time. Nayak, 228-29.

119. For the IDS to monitor traffic "transparently," it would have been obvious to a POSITA for the IDS to perform its content determinations "*in real time*" because delaying traffic would interfere with communication sessions. In other words, for the IDS to operate "transparently," it would need to make content determinations "*in real time*."

120. Burns and Yang describe identifying an application associated with a packet with an application identification module 51 within stateful inspection engine 28. Burns, 8:11-23, Fig. 3; Yang, 9:27-49. Thus, application identification module 51 is part of the IDS's "forwarding plane" that operates transparently. As part of the forwarding plane, it would have been obvious for the application identification module 51 to operate "*in real time*" as claimed to facilitate "*continued operation by the user's device*."

121. The forwarding plane of a network device is responsible for forwarding packets toward their destination with minimal delay. U.S. Patent Publication No. 2005/0281192, [0038] (“excessive delay” is a “forwarding plane failure”); U.S. Patent Publication No. 2011/0110328, [0030] (“achieving lower latency” is a forwarding plane advantage); U.S. Patent Publication No. 2013/0329578, [0029] (describing forwarding plane architectures “to reduce latency and other delays”). It would have been obvious to a POSITA that an IDS functioning in a “transparent” manner is undetectable to the user and is thus occurring in *real-time*, as, “[m]ost commercial network intrusion detection systems run in real-time.” Proctor, 38.

122. Since Burns’s IDS monitors traffic between enterprise computers (node 8) or security management device (18) (“*user’s device*”), it would have been obvious for the packets analyzed by the IDS to be passed through the IDS “*during communication.*” See [17.0], Burns, Fig. 1. As such, a POSITA would have found it obvious for data traffic to pass “*between the remote server and the user’s device.*”

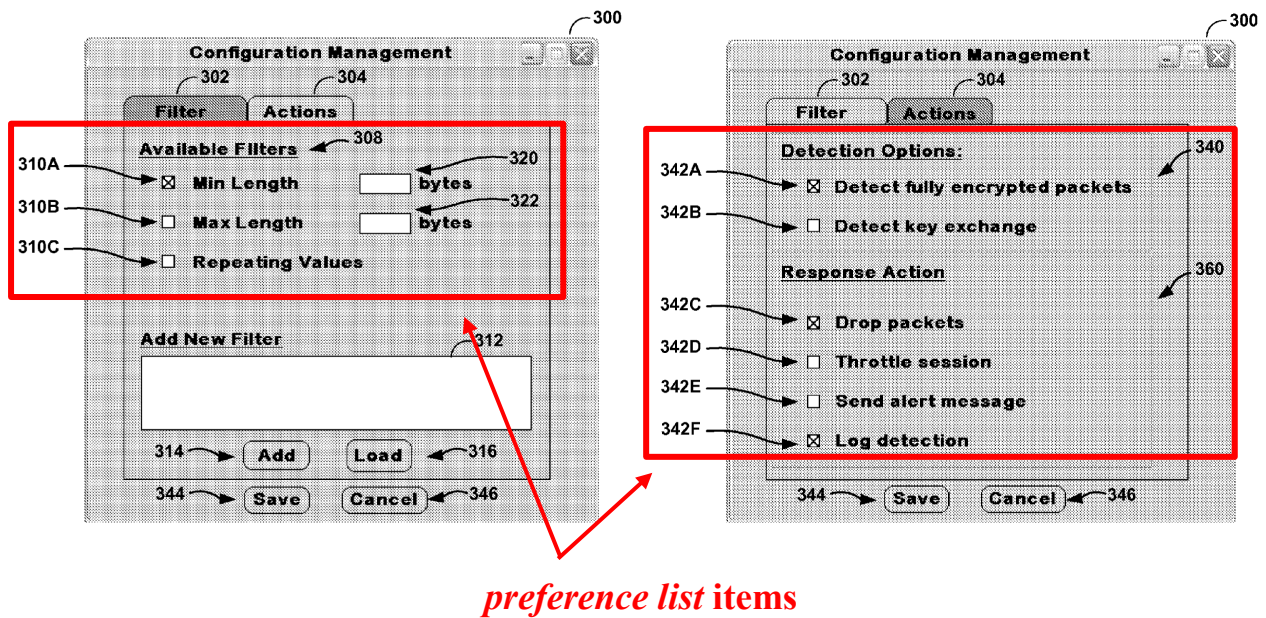
2. Claim 18

a. [18.0] *The system of claim 17, further comprising*

123. Limitation [18.0] would have been obvious for the reasons discussed in [17.0].

b. [18.1] *a communication data pattern database, coupled to the processor and*

124. As discussed at [17.3], the “security management module 44 receives configuration information from administrator 42,” and can store information in a “database, a text file, or any other appropriate structure.” Burns, 9:64-65, 20:65-67. Burns describes storing other “information that administrator 42 may find useful” relating to detected attacks in a database. Burns, 20:64-65. Therefore, it would have been obvious to store the administrator’s configuration for *data patterns* in a “database.” Options for Burns’s administrator configuration are seen in Figs. 7A-7B, where an administrator can selectively enable or disable certain preferences, such as whether to detect repeating values or an encryption key exchange. Burns, 17:58-20:67.



Burns , Figs. 7A-7B (annotated)

125. As addressed at [17.8], Burns describes that the administrator's configuration information may also be used to specify "patterns or other attack definitions" to instruct the IDS to reject or allow packets based on certain data patterns, including (1) size, (2) encryption key exchange, and (3) repeating sequences or values. Burns also describes patterns and anomalies that are associated with known network attacks. Burns, 8:6-10, 10:52-55.

126. As addressed at [17.8], Burns describes that a packet's size may be used to determine how to classify a packet. *See* Burns, 2:34-35, 14:17-19, 14:42-43.

127. As addressed in the analysis of [17.8], Burns describes that whether an encryption key is exchanged can be used to determine the "*pattern*" of the data packet. A POSITA would have recognized that different key exchange information (*data patterns*) could be associated with different protocols. *See, e.g.* Burns 12:47-13:8 (describing SSL key exchange patterns that require a specific order of packets sent with ClientHello, ClientMasterKey, and ServerVerify messages to exchange keys).

128. As addressed in the analysis of [17.8], Burns describes that a "repeating pattern," "repeating value," or "repeating sequence" can be identified in one or more data packets. A POSITA would have found it obvious that these repeating sequences were "*communication data pattern[s]*." Burns, 14:32-35

(“identifying repeating patterns of varying lengths”), 18:36-38 (“encryption detection module 58 compares data in the packet to other data in the packet to find a repeating sequence”), 18:43-44 (“IDS 20 may utilize a repeating values filter”).

129. Furthermore, Burns describes that patterns may be specified as “attack definitions” for the “attack detection module 52.” Burns, Fig. 3. Burns describes that patterns table 54, attack definitions 33, and anomalies table 56 are used by the stateful inspection engine. Burns, 7:62-64. A POSITA would have appreciated that the “attack detection module” would detect attacks based on the configurable attack definitions.” Yang provides further details about how anomaly and pattern information may be used, “stateful inspection engine 28 combines pattern matching with protocol-specific anomaly analysis to detect sophisticated attack behaviors.” Yang, 11:49-51. These tables would correspond to a “*communication data pattern database*.”

130. For the reasons addressed at [17.7], it would have been obvious that a “*communication data pattern database*” would be coupled to the “*processor*,” as addressed at [17.6], to instruct the IDS to act accordingly based on the administrator’s configuration information. Therefore, the prior art renders obvious a “*communication data pattern database, coupled to the processor*.”

c. **[18.2] comprising at least one data pattern corresponding to sharing of at least one data packet from the user's device,**

131. For the reasons addressed at [17.8] and [18.1], it would have been obvious that packets which the IDS determined to be in accordance with the administrator's configuration (containing allowed "*data patterns*") would be shared. A POSITA would have recognized that this would involve "*sharing of at least one data packet from the user's device,*" because TCP can be a bi-directional protocol as described in Yang. Yang, 14:26-29 ("IDS 20 detects the response packet flow and associates the response with the initial request, thereby defining a bidirectional communication session.")

d. **[18.3] wherein the processor is configured to modify data packets corresponding to requests for retrieval of data packets and communication data types that are not compatible with communication data patterns from a communication data pattern database.**

132. Limitation [18.3] contains inverse elements of limitation [17.8]. Whereas limitation [17.8] involves a processor instructing the remote server to determine a data type or alternatively, both a data type and a data pattern, that is "*compatible with allowed patterns for sharing,*" limitation [18.3] involves a "*processor configured to modify....data packets and communication data types that are **not** compatible with communication data patterns.*"

133. As addressed at [17.6] and [17.7], it would have been obvious for the claimed "*processor*" to be part of the IDS (*remote server*) as a processor acts as the

“brain” of the system. Since the IDS is responsible for detecting whether communications correspond with the preferences, the prior art renders obvious “*wherein the processor is configured to....*”

134. As discussed at [17.8], Burns discloses that some “*communication data types*” and “*communication data patterns*” allowed. A POSITA would have recognized that Burns’s IDS is also an IDPS and would therefore also prevent possible network intrusions. This is in accordance with the common knowledge that an IDS compares administrator-specified patterns or other known attacks with the packet’s identified application to determine which actions to take with the packet. *See, e.g.,* Nayak, 229-33.

135. Burns and Yang describes “attack behaviors,” as part of “attack definitions 33.” A POSITA would have recognized that attack behaviors would correspond to *data patterns* that are not wanted or pose some kind of threat to the network. It would have been obvious that the administrator’s configuration of the IDS would be used to prevent packets representing unwanted behaviors, rendering obvious “*data packets...not compatible with communication data patterns from a communication data pattern database.*”

136. Burns describes that packet flows may “collectively form a single communication session between a client and server.” Burns, 7:9-15. Therefore, if a packet is detected flowing from the client or the server that does not correspond with

the administrator's configuration, Burns describes that the IDS takes action. Burns gives the examples of (1) dropping the packet, (2) automatically closing the communication session, or (3) throttling the communication session in response to a packet that does not match the administrator's configuration:

In addition, stateful inspection engine 28 may take additional actions, such as dropping the packets associated with the communication session, automatically closing the communication session, or other actions.

Burns, 7:47-51.

For example, administrator 42 may configure IDS 20 to block all fully encrypted packets and log information about the communication session associated with the packet. Administrator 42 may also configure IDS 20 to throttle down the communication session associated with the packet to minimize the bandwidth used by the communication session. Administrator 42 may also configure IDS 20 to block future communication sessions from either or both of the communicating devices.

Burns, 11:50-58.

137. (1) Dropping (or “blocking”) the packet: Burns describes that in response to an unwanted packet or potential attack, one form of modification is “blocking” the packet:

In addition, stateful inspection engine 28 may take additional actions, such as dropping the packets associated with the communication session...

Burns, 7:47-49.

138. This is consistent with the '824 patent that explains that “blocking the packet” is one form of modification.’824 patent, 2:44-50.

139. (2) Automatically closing the communication session: A POSITA would have understood TCP protocols and would have known that in order to close a TCP session, a packet is “*modif[ie]d*.” To close a TCP connection, the device that wishes to close the connection will send a FIN flag in the packet header. The other device will respond by sending an ACK, followed by its own FIN message. Kozierok, 763. In a situation where one device wishes to end the connection, but the other is unresponsive, the device may send an RST flag in the packet header. Kozierok, 760. Such modifications were well known to a POSITA:

[A] TCP endpoint that desires to initiate session closing operations sends a FIN command to the other endpoint. A FIN is represented by a TCP header flag.

U.S. Patent Publication No. 2006/024858, [0095].

A TCP session is closed by both sides sending packets comprising the FIN flag indicating that the sender has no more data to send.

U.S. Patent Publication No. 2004/0013112, [0109].

140. (3): Throttling the communication session: Burns describes that the IDS may “throttle down the communication session associated with the packet.” Burns, 11:53. A POSITA would have recognized that one way to throttle a session would be to adjust the TCP window size by “*modifying*” the TCP header. Kozeriok, 805-06.

141. Therefore, the IDS blocking, automatically closing, or throttling the communication session when a packet does not correspond with the data patterns specified in the administrator’s configuration renders obvious “*wherein the processor is configured to modify data packets corresponding to requests for retrieval of data packets and communication data types that are not compatible with communication data patterns from a communication data pattern database.*”

3. Claim 19

- a. **[19.0]** *The system of claim 18, wherein data packets from the user's device are selected from the group consisting of user's device files, user's device characteristics, user's device indirect attributes, user's device sensor data, user's, user's device form data, user's device dynamic memory and user's device static memory.*

142. I understand from Cisco’s counsel that a recitation of each element listed after the colon is not necessary to be shown in the prior art. Nevertheless, my analysis below demonstrates multiple components listed in this claim.

143. As described at [17.0], the “computing node” 8A corresponds to a *user’s device*. Burns states that private computing devices represented by nodes

represent “workstations, file servers, print servers, database servers, printers and other devices.” Burns, 4:32-35. It would have been obvious to a POSITA that a “file server” would contain “*user device files*” that could be sent as packets.

144. Burns and Yang both disclose “HTTP traffic,” which a POSITA would have understood to be associated with web data. Burns, 12:13-15, 10:38-39 (“port 80 is typically associated with HTTP traffic...”); Yang, 5:8-12 (“a single TCP connection can be used to send (receive) multiple HyperText Transfer Protocol (HTTP) requests (responses).”), 5:39-40 (“Example anomalies for the HTTP protocol include missing HTTP version information...”); Kozierok, 1320 (“HTTP is the TCP/IP application layer protocol that implements the Web, by enabling the transfer of hypertext documents and other files between a client and server.”); U.S. Patent Publication No. 2002/0143897, (“The browser communicates with the server using standard networking protocols, the most common of which is Hypertext Transport Protocol (HTTP).”); U.S. Patent Publication No. 2007/0248105 (“HTTP protocol used between a WWW (World Wide Web) server and a Web browser when browsing a webpage”). Therefore, it would have been obvious that the *data packets* analyzed by the IDS would include the claimed “*user’s device browser data*”.

145. Yang describes that a “compound attack definition” can be created to identify a protocol (to filter) and cites “HTTP form data” as a type of context an administrator may filter for. Yang, 15:35-41. Therefore, it would have been obvious

that the claimed “*data packets*” analyzed by the IDS would include “*user’s device form data.*”

4. Claim 20

- a. [20.0] *The system of claim 18, wherein at least the communication module and the processor are embedded on a single hardware component.*

146. As addressed at [17.5], the IDS contains “forwarding component 31,” which a POSITA would have understood to correspond to the “*communication module.*” As addressed at [17.6], a POSITA would have found it obvious that the claimed “*processor*” would instruct Burns’s IDS. A POSITA would have recognized that IDS devices are often implemented using hardware. Nayak, 225. Burns describes that its methods “may be performed in hardware, software, or any combination thereof within a network device.” Burns, 21, 14:16. A POSITA would have therefore found it obvious to combine the processor that instructs the IDS and the IDS in the same hardware component. Therefore, it would have been obvious for Burns’s IDS to exist as a “*single hardware component.*”

X. THE COMBINATION OF BURNS, YANG, AND WITTENBERG RENDERS OBVIOUS CLAIMS 1-16

A. Summary of Wittenberg

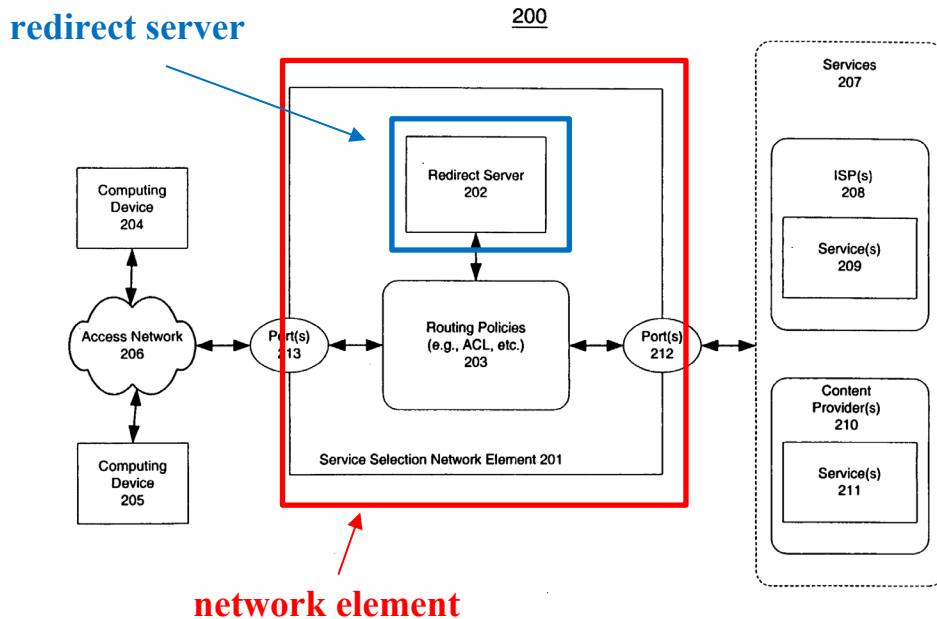
147. Like Burns and Yang, Wittenberg describes directing network traffic containing packets based on a set of routing policies. Wittenberg is focused on

providing a redirect to authenticate a device that seeks to join a network. Wittenberg, [0016]-[0017].

148. Wittenberg describes that a computing device will send a request to access a given service through “network element 201” Wittenberg, [0029]. Before routing the computing device to the given service, the network element examines the packet header to determine whether the packet is an HTTP packet:

That is, when network element 201 receives a packet from one of the computing devices 204 and 205, network element 201 examines the header of the packet, such as TCP/IP header of the packet, to determine whether the packet is an HTTP packet. Whether the packet is an HTTP packet may be determined based on conventional use of ports for the HTTP packets. In one embodiment, a packet is an HTTP packet when its destination port of the TCP header is directed to port 80.

Wittenberg, [0029].



Wittenberg, Fig. 2A (annotated)

149. The network element accesses the routing policies 203 to determine whether the packet should be routed to a login screen or other reroute definition based on context information. Wittenberg, [0029]. Alternatively, all HTTP requests can be redirected. Wittenberg, [0029].

150. Once it is determined that a packet should be redirected based on the redirect policies, Wittenberg describes that the packet can “be forwarded, via an internal logical interface, to redirect server 202 without invoking an external dedicated redirect server.” Wittenberg, [0030]. The redirect server can then forward the redirect URL “back to the browser of the computing device,” and the computing

device may “access the redirect destination, via network element 201.” Wittenberg, [0030]. These processes are all performed within network element 201.

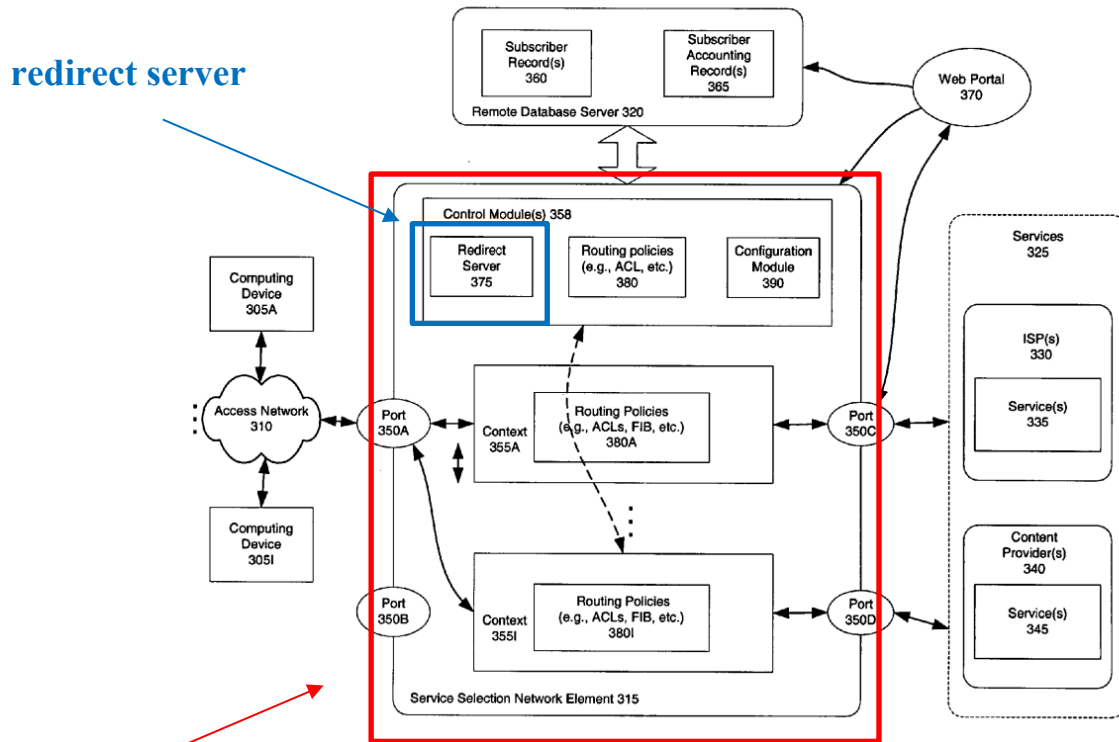


Fig. 3

network element

Wittenberg, Fig. 3 (annotated)

151. The network element contains different “contexts,” and for each context, the network element can redirect the user to a different website via the “redirect server.” Wittenberg, [0040]-[0048]. As one example, the redirect server presents the user with a subscriber login each time they access a given website. Wittenberg, [0056], Fig. 7. Each time a user reconnects, a web form appears where

they must enter their username and password. Wittenberg, [0046] Fig. 7. Wittenberg additionally describes that a similar web form could be used when a new subscriber tries to connect, or when a subscriber tries to input information such as payment method, contact information, etc. Wittenberg, [0046]. These windows and their configuration may be presented on a separate web portal 370, but the web portal may “be maintained within the service selection network element 315.” Wittenberg, [0047]. Once the redirect screen is completed, the user may continue to access the web and the services they were initially seeking to access. Wittenberg, [0033].

B. Reasons to Combine Wittenberg, Burns, and Yang

152. Like Burns and Yang, Wittenberg is directed to managing network traffic. Whereas Burns and Yang are directed towards detecting and blocking incoming traffic that does not conform with an administrator’s policies, Wittenberg is directed towards a redirect system that can be used to verify or authenticate a user. A POSITA would have combined the disclosure of Wittenberg with Burns and Yang create a unified system that requires that a user be authenticated before their device joins a network.

153. Burns and Yang describe an IDS which has both detection and prevention functionality for attacks and anomalies (also referred to as an IDPS). A POSITA who wanted to prevent attacks would have recognized that an attack could come from outside of the network or from inside the network if a user’s device was

compromised. These attacks could be in the form of an unauthorized user attempting to access the network, or in the form of a virus or bot. By presenting an authentication screen that requires a user enter their username and password into a web form, Wittenberg's disclosure would be valuable for an enterprise system such as the one described in Burns because it would authenticate that a user is an authorized user rather than an unauthorized third party. A web login approach would have been a well-known and obvious way to achieve this functionality. *See, e.g.*, U.S. Patent Publication No. 2008/0016570, claim 15 ("web-based visualization interface that facilitates configuration of the system and forensic analysis of captured attack information by administrators"); [0047] ("secure configuration and administration may be provided... through an HTTPS (Secured HyperText Transfer Protocol-Secure Sockets Layer (SSL) enabled) Web Browser."); U.S. Patent No. 7127743, 6:39-40 ("configuration tables accessible through web client interface"); U.S. Patent Publication No. 2006/0179472, [0099] ("System changes are typically accomplished by authenticated administrators that access system 100 through the World Wide Web."); U.S. Patent Publication No. 2004/0187028, [0004] ("Using a conventional web browser, a system administrator can browse to the address of a particular device. The embedded web server returns a web page allowing the administrator to select configuration settings for that device").

154. Once a user is authenticated, Burns's IDS would then analyze the network traffic. The combined system would use Yang's TCP port number method to determine if a packet represents an allowed application or protocol, and would compare the traffic to the administrator's configuration to ensure that only allowed traffic is permitted.

155. A POSITA who wants to improve enterprise network security would have recognized that malicious activity could come from outgoing or incoming traffic. In order to supplement the IDS of Burns and ensure that only desired users are able to access the system, it would have been obvious to use a known technique (login webpage) to improve Burns's IDS by confirming the identity of a user through their username and password. The combination of these systems would have achieved a predictable result, as both techniques were well-known and both would serve the purpose of improving security for a similar device. Therefore, it would have been obvious to incorporate the system and methods of Wittenberg into the system and methods of Burns and Yang.

C. Detailed Analysis of Claims

1. Claim 1

- a. **[1.0] *A method of dynamic management of private data during communication between a remote server and a user's device, the method comprising:***

156. To the extent that the preamble is limiting, Burns and Yang disclose it, as addressed at [17.0].

157. In the combined system of Burns and Wittenberg, once a user has accessed a network (as further described in [1.1] and [1.2]), a communication session can be established between the *user's device* and a *remote server*.

158. Burns describes “blocking unidentified encrypted communication sessions.” Burns, Abstract. Burns further discloses that “all or portions of an application-layer header within each packet” may be encrypted. Burns, 5:3-5. A POSITA would have recognized that encrypting a packet would have prevented a third party who did not have the encryption key from reading it. Nayak, 163-67. Because someone encrypting data would likely be trying to keep that data private, Burns renders obvious “*private data*.” This interpretation is supported by the Ferreira reference cited by the '824 patent, which states:

The article presents a solution for storage and **management of private data**... The approach is based on a middleware architecture supported by homomorphic **encryption techniques** combined with dynamic indexing mechanisms.

Ferreira, Abstract.

159. As Burns discloses identifying and terminating a session in response to the receipt of an encrypted (*private*) data packet, a POSITA would have understood Burns to teach “*a method of dynamic management of private data*.” Burns, 11:30-38.

160. In Fig.1, Burns that the “intrusion detection system” (IDS) “attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices in public network 6.” Burns, 4:36-39.

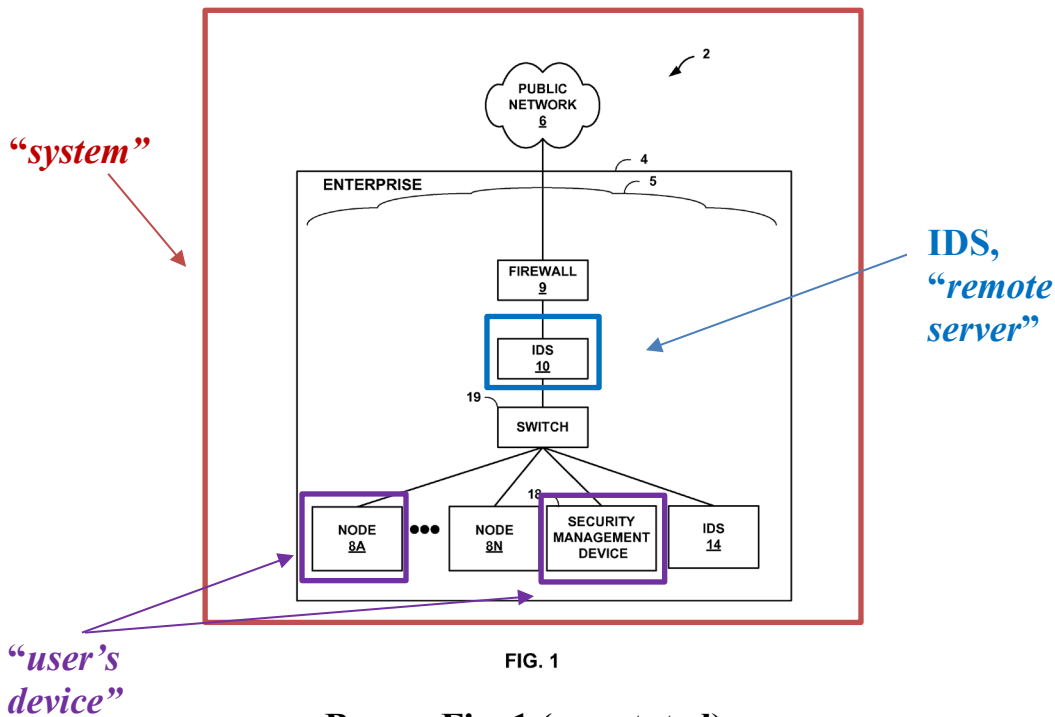


FIG. 1

Burns, Fig. 1 (annotated)

161. A given node, computing device or workstation corresponds to the claimed “a user’s device.” Burns, 4:32-34.

162. The IDS of Fig. 1 is separated from the *user’s device* by the switch 19. A POSITA would have understood the IDS to correspond to a “remote server.” See,

e.g., Nayak 238-42 (showing different configurations of a switch separating an IDS from devices).

163. Burns further explains that its IDS “attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices.” Burns, 4:37-38. Therefore, a POSITA would have appreciated that Burns teaches “*communication*” between the IDS (*remote server*) and the nodes (*user’s device*).

b. [1.1] *receiving, by the user’s device, a request for retrieval of at least one data packet from the user’s device,*

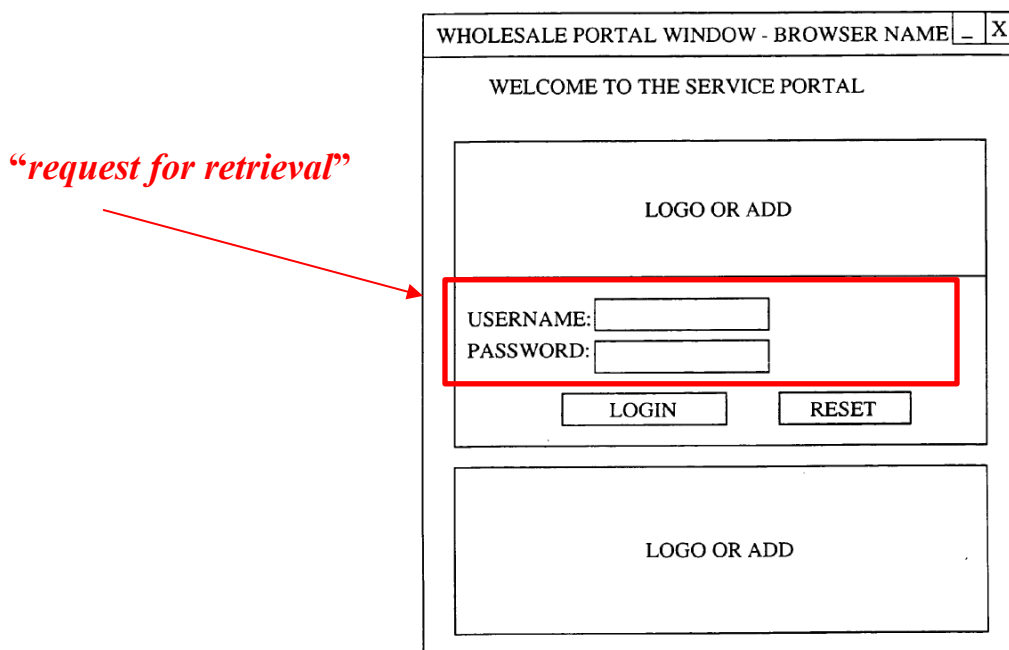
164. A POSITA would have recognized that in addition to an IDS that protects from third-party intrusions on a network, devices within a network could also pose threats to the network. These intrusions could come in the form of a virus or bot, or by an unauthorized user.

165. As addressed at [19.0], Burns and Yang disclose “HTTP traffic and HTTP form data,” which a POSITA would have recognized as web data and web form data. Burns, 10:39; Yang, 15:35-41. A POSITA would have recognized that HTTP form data may prompt a user to input information onto a website, such as a username, password, address, or payment information.

166. Wittenberg describes that a user may attempt to access certain content online. Based on the network’s “routing policies,” the user may be rerouted to another website to input their login details into a web form. Wittenberg, [0047], Fig.

8. Wittenberg describes that “pop-up windows” may also collect “additional information.” Wittenberg, [0046].

167. The web form corresponds to a “*request for retrieval*,” as it prompts the user to input information into the form fields. The user may then enter their username and password, which would be provided back to the web server as HTTP form data, corresponding to the “*at least one data packet*.” Therefore, the combination of Burns and Wittenberg render obvious “*receiving, by the user’s device, a request for retrieval of at least one data packet from the user’s device*.”



Wittenberg, Fig. 8 (annotated)

c. [1.2] wherein the user's device is configured to provide a response corresponding to the received request;

168. In a combined system of Wittenberg and Burns, the user, using the user's device, would fill out the web form as shown above in Wittenberg, Fig. 8, and fill in the fields as required to provide a username and password, then click "LOGIN" (provide a response corresponding to the received request). It would have been obvious to a POSITA that the user's device would be "configured to provide a response corresponding to the received request," because filling out a web form is a common task for the "private computing" devices in Burns. Burns, 4:32-35. Once a user has filled out the web form as described in Wittenberg, the user's device would be connected to the system as described in Burns at Fig. 1.

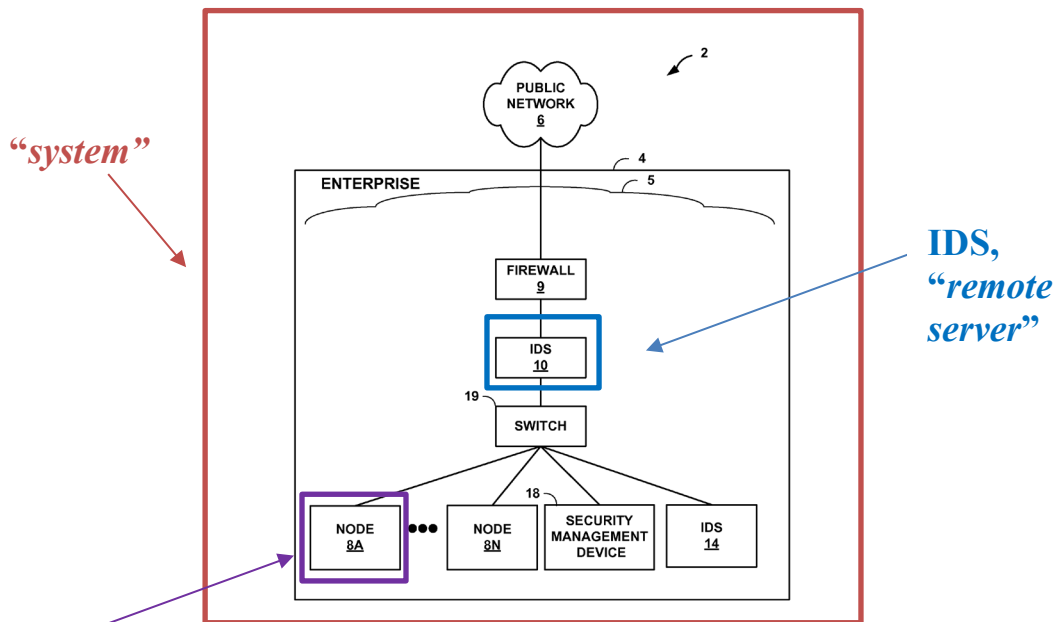


FIG. 1

Burns, Fig. 1 (annotated)

- d. [1.3] *determining, by the remote server, at least one communication data type of the at least one data packet corresponding to the received request,*

169. This limitation is similar to the limitation [17.8] and would have been obvious for the same reasons. A POSITA would have understood that a “*data type*” as in claim 17, and a “*communication data type*” as in claim 1 both refer to an application or protocol as described in Burns.

170. As addressed at [1.0], Burns’s IDS corresponds to the *remote server*. The IDS identifies, i.e., *determines*, the applications and protocols for communication sessions that include “*at least one data packet*”:

In one embodiment, IDS 10 attempts to identify applications and protocols for each communication session between computing nodes 8 and other computing devices in public network 6.

Burns, 4:36-39.

171. As addressed at [1.1] and [1.2], the information input by the user into the web form would correspond to the “*at least one data packet*.” Once connected to the system, the IDS described in Burns and Yang would function as described below to monitor network traffic for potential attacks or anomalies.

172. Burns references Yang’s disclosure for greater detail regarding “[e]xemplary techniques for identifying specific applications and protocols.” Burns, 4:36-42. Yang describes identifying the “type of application and protocol associated

with the packet flow” through static port mapping with an “application identification module.” Yang, 9:59-66. Yang’s application identification module uses a packet’s TCP port number to identify certain applications and/or protocols. Yang, 9:59-10:24, Table 1. Yang’s “port number” corresponds to a “*characteristic*” of the packet, and the “application” corresponds to a “*communication data type*.”

TABLE I

PORT	APPLICATION
20	FTP
22	SSH
23	Telnet
25	SMTP
43	WHOIS
53	DNS
67	BOOTP or DHCP
70	Gopher
79	Finger
80	HTTP
109	POP
110	POP3
113	ident/IRC
118	SQL
119	NNTP
194	IRC
443	HTTPS
445	SMB
564	RTSP

Table 1, Yang

173. A POSITA would have known that Yang’s “static port mapping” approach would identify the type of data contained in the packet, because port numbers (*characteristics*) are standardized to correspond to certain applications and protocols. Kozierok, 708-09. The ’824 patent states that a *data type* can be

determined by “verifying at least one characteristic of the data packet.” ’824 patent, 8:26-28.

174. For example, for the port value 80 (*characteristic*), the *data type* is HTTP. A POSITA would have known that the *data type* HTTP, or “Hypertext Transfer Protocol (World Wide Web),” would indicate that the packet’s contents are website data. Kozierok, 709. Similarly, for the port value 443, the *data type* HTTPS would inform a POSITA that the packet content is encrypted website data. Kozierok, 709. For the port value 109, the *data type* POP “Post Office Protocol” would indicate that a packet contains email data. Kozierok, 709. *See also* Kozierok, Table 43-1 (showing well-known port numbers and their corresponding applications); RFC 1700, (listing port numbers for thousands of known applications.)

175. In the combination of Burns, Yang, and Wittenberg, once a user’s device is connected to the system, Burns’s IDS (*remote server*) would receive a packet, and Yang’s application identification module would use static port mapping to identify the packet’s port number and its corresponding application or protocol (*data type*). Burns describes an IDS that functions bi-directionally and would therefore be able to detect and protect against potentially malicious packets coming from either a user or a third-party. Burns, 8:60 (“the default protocol decoder attempts to analyze the bi-directional communications for the communication session to identify a key exchange for cryptography.”)

176. Because Burns's IDS (*remote server*) identifies an associated application or protocol (*data type*) based on mapping port numbers from TCP packet headers (*characteristics*), the combination of Burns and Yang render obvious "*determining, by the remote server, communication data type of the at least one data packet corresponding to the received request.*"

- e. **[1.4] *wherein the at least one communication data type is determined in accordance with characteristics of the communication data packet, and***

177. As addressed above in the analysis of [1.3], the port number (*characteristic*) approach in Yang can be used to identify an associated protocol or application (*data type*), rendering obvious this limitation.

- a. **[1.5] *wherein the content of the at least one data packet is not read by the remote server for continued operation by the user's device in real time during communication between the remote server and the user's device;***

178. Limitation [1.5] is substantially identical to limitation [17.10], therefore it would be obvious for the same reasons as analyzed above. As addressed at [17.10], Burns and Yang describe an IDS that "includes 20 a forwarding plane 22 that transparently monitors inbound network traffic 24 and forwards the network traffic as outbound network traffic 26." Burns,6:30-34. It would have been obvious to a POSITA that an IDS functioning in a "transparent" manner is an IDS that is in "*continued operation with the user's device.*" A "transparent" approach could also be said to occur in *real-time*.

179. As explained in the analysis of [1.1] and [1.2], Burns's IDS (*remote server*) receives a communication (*data packet*) that is sent from node 8 (*user's device*). Burns's IDS monitors this traffic and determines an application or protocol corresponding to the traffic (*a content*) of the communication, rendering this limitation obvious.

b. [1.6] *receiving, by the user's device, a privacy preference for the user's device,*

180. As discussed in the analysis of [17.3], Burns's IDS can be tailored to an administrator's desired configuration information. Burns, 1:21-35. Burns allows a system administrator to "configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a specified list of identifiable applications, or prevent all communications." Burns, 5:30-34. Furthermore, an administrator can specify attack definitions and desired responses:

Initially, security management module 44 receives configuration information from administrator 42 and, in response, configures IDS 20 to monitor a network or portions thereof (subnets) of interest... configuration manager 44 presents a user interface by which **administrator 42 specifies patterns or other attack definitions 33**. For example, administrator 42 **may configure IDS 20 to block all packets deemed to be fully encrypted and to log information about the communication session associated with the blocked packets**. Administrator 42 **may also configure IDS 20 to throttle down (i.e.,**

bandwidth limit) the communication session associated with the packets to minimize the bandwidth used by the communication session.

Burns, 9:64-109.

181. The administrator's configuration information allows the administrator to explicitly permit certain applications, deny others, and provide an "attack definition." A POSITA would have appreciated that, because the configuration information allows administrator to tailor the IDS to their preferences, the configuration information specified by the administrator corresponds to a "*preference list.*"

182. Burns's administrator is not shown in Fig.1, which shows node 8A that corresponds to a "*user's device.*" Burns explains that the system includes "internal computing nodes" (8A-8N) that correspond to a "*user device.*" A POSITA would have understood, in view of Burns's background discussion, that these "internal computing nodes" would include "end user computers." Burns, 1:8-16. Burns further explains that an administrator may be a user. Burns, 6:5-8 ("a user, such as a system administrator"), 9:56 ("a user, such as administrator"). Therefore, based on the combined disclosure, a POSITA would have understood that an "administrator" would use an internal computing node as a user.

183. Therefore, the administrator's configuration information for the IDS renders obvious "*by the user's device, a privacy preference for the user's device.*"

- c. **[1.7] wherein the privacy preference comprises a list of allowed data packet communication types for sharing during communication;**

184. As addressed at [17.4], Burns's IDS can be configured to provide security according to an administrator's desired configuration information:

Where IDS 10 is able to identify the application using a particular communication session, IDS 10 may **either permit or prevent** the communication session from continuing. For example, a system administrator may configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a **specified list of identifiable applications**, or prevent all communications except for communications from a **specified list of permitted software applications**.

Burns, 5:27-34.

[A]dministrator 42 may configure IDS 20 to take particular actions when a fully encrypted packet is identified. For example, the user interface may permit the administrator to **drop all fully encrypted packets**, log details about the communication session associated with the fully encrypted packets, drop the packets only if a key exchange has not been previously identified for the communication session, throttle down the communication session with fully encrypted packets to reduce the bandwidth usage of that communication session, or other actions.

Burns, 6:47-56; *see also* Burns, 12:7-26 & 13:9-57.

185. Burns describes that the administrator's configuration allows packets containing certain applications to be forwarded, while others are modified, which

renders obvious “*a list of allowed data packet communication types for sharing during communication.*”

- d. [1.8] *modifying data packets corresponding to requests for sharing of responses that are not compatible with the received privacy preference; and*

186. As discussed at [18.3], an administrator of Burns’s IDS could specify patterns or attack definitions that are “allowed” to be shared, and those that are not allowed:

For example, a system administrator may configure IDS 10 to explicitly **allow all identifiable applications**, allow all applications except for a specified list of identifiable applications, or **prevent all communications except for communications from a specified list** of permitted software applications.

Burns, 5:29-34.

187. More specifically, in Burns, the administrator configures a “specified list of permitted software applications,” and the IDS “*compar[es]*” the packet’s identified application (*data type*) to the applications on the administrator’s list (*the preference list*). Burns, 5:34. This disclosure is consistent with common knowledge that an IDS compares administrator-specified patterns or other known attacks with the identified application to determine which actions to take with the packet. *See, e.g.,* Nayak, 229-33.

188. Once Burns's IDS compares the packet's identified application with the applications in the administrator-specified list, the IDS acts accordingly by, dropping the packet, closing the communication session, or throttling the communication:

In addition, stateful inspection engine 28 may take additional actions, such as dropping the packets associated with the communication session, automatically closing the communication session, or other actions.

Burns, 7:47-51.

Administrator 42 may also configure IDS 20 to throttle down the communication session associated with the packet to minimize the bandwidth used by the communication session.

Burns, 11:53-55.

189. Burns's IDS is bi-directional and therefore can monitor traffic between a client and a server. Burns, 7:8-15 ("packet flows that collectively form a single communication session between a client and a server."). Therefore, it would have been obvious for Burns's IDS to analyze and subsequently modify packets containing a web form such as the login webpage described in Wittenberg. Wittenberg, Fig. 8. A POSITA would have recognized that a response to this web form would correspond with "*data packets corresponding to requests for sharing of responses.*"

190. Therefore, A POSITA would have recognized that if a user input information into a web form that did not correspond to the administrator's

configuration information, Burns's IDS would act accordingly to drop a packet, terminate a session, or throttle the communication. The prior art therefore renders obvious "*modifying data packets corresponding to requests for sharing of responses that are not compatible with the received privacy preference.*"

- e. **[1.9] *maintaining communication between the remote server and the user's device, with sharing of the modified data packets.***

191. Burns's IDS includes the stateful inspection engine that "instructs forwarding component 31 to forward the packet flows." Burns, 7:53-55.

192. As discussed in the analysis of [1.9], Burns's disclosure of "dropping a packet" corresponds with "*modifying data packets corresponding to requests for sharing of responses that are not compatible with the received privacy preference.*" When the IDS (*remote server*) drops a packet, the packet would still be "*shar[ed]*," because the packet containing the modified header with either a FIN or RST flag is sent by the IDS to indicate that the IDS wishes to end the connection. Communication is "*maintain[ed]*" between the IDS (*remote server*) and the *user's device* until the FIN flag is returned by the *user's device* and the *remote server* sends an ACK in return.

193. Alternatively, when the IDS (*remote server*) drops a packet, communication is still "*maintain[ed]*" and the packet is still "*shar[ed]*" because the IDS "log[s] information about the communication session associated with the

packet.” Burns, 11:51-52. A POSITA would have known that this packet information would need to be shared to storage, as it would enable the IDS to identify and block associated abnormalities sent in the future. Nayak, 230.

194. Therefore, Burns renders obvious “*maintaining communication between the remote server and the user’s device, with sharing of the modified data packets.*”

2. Claim 2

- a. [2.0] *The method of claim 1, wherein the communication data type of the at least one data packet is determined from metadata of communication with the remote server.*

195. As discussed in the analysis of [1.3], Burns describes that a packet’s *data type* (application or protocol) may be identified based on the TCP port value. *See also* Kozierok, 708-09. As discussed at [1.3], a *data type* of a packet can be determined based on a *characteristic* of a packet. For example, the *characteristic* port value 80 corresponds to the *data type* HTTP, which would indicate that a packet contains website data. It would have been obvious to a POSITA that a TCP port value in a packet header is “*metadata.*”²

² A TCP port number was commonly considered to be a form of “metadata” about a packet. *See* U.S. Patent Publication No. 2017/0230065, [0030] (“traffic metadata, such as MAC address, IP address, TCP port, UDP port, etc.”); U.S. Patent

3. Claim 3

a. [3.0] *The method of claim 1, further comprising:*

196. To the extent the preamble is limiting, the limitation [3.0] is disclosed by Burns, for the same reasons as addressed in [1.0].

b. [3.1] *determining, by the user's device, at least one communication data pattern of at least one data packet corresponding to the received request; and*

197. As addressed in the analysis of [17.8] and [18.1]-[18.3], Burns and Yang disclose multiple methods for determining “*allowed patterns of data packets*” and “*communication data patterns.*” Burns describes that a pattern may consist of size, encryption key exchange, or repeated sequences or values. *See, e.g.* Burns, 19:30-33 (“instructs stateful inspection engine 28 to analyze incoming packets to determine whether those packets represent a key exchange.”), 14:32-35 (“encryption detection module 58 compares data in the packet to other data in the packet to find a repeating sequence.”)

Publication No. 2014/0351106, [0033], (“a port number (e.g., a Transmission Control Protocol (TCP) port number) may also be included in the metadata record”); U.S. Patent Publication No. 2012/0230208, [0057] (“metadata extracted from each outgoing TCP packet may include source and destination IP addresses, source and destination TCP ports”).

198. Both Burns and Yang disclose that the identification of “*data patterns*” is conducted by aspects of the IDS (*remote server*). *See, e.g.* Burns, Figs. 2 and 3 (showing that the IDS contains the stateful inspection engine 28, and that the encryption detection module 58 is a component of the stateful detection engine.) In Burns, the IDS is represented as an element of a larger enterprise network. *See* Burns, Fig. 1, 4:25-26 (“In the example embodiment of FIG.1, IDS 10 is a single network device.) However, Burns also describes that the IDS may be configured differently within the network:

In some embodiments, enterprise network 5 includes multiple IDSs 10 and 14 located within different regions (e.g., sub-networks) of enterprise network 5. Security management device 18 may operate as a central device for managing IDSs 10 and 14. **Although the example illustrated in FIG. 1 is described in terms of dedicated IDSs 10 and 14, the functionality described herein may be incorporated within other devices**, such as firewall 9 or switch 19.

Burns, 6:9-17.

199. A POSITA would have recognized that the configuration shown in Burns is a network-based IDS. Nayak, 228. It would have been obvious to a POSITA that Burns’s statement “the functionality described herein may be incorporated within other devices” would include a host-based IDS. Burns, 6:15-17. A host-based IDS would involve software, or another agent installed on the endpoint device to monitor and report application activity. Nayak, 227.

200. A POSITA would have recognized that a system could consist of both network-based and host-based IDSs:

IDS' normally fall into a number classifications. These classifications include network-based, host-based, protocol-based, and application based intrusion detection systems. **Combinations of these classifications are common. These combinations, also known as hybrid intrusion detection systems, including, for example, a combination of network-based and host-based intrusion detection systems.**

U.S. Patent Publication No. 2009/0254970, [0005]; *see also* Nayak, 228.

201. Therefore, a POSITA would have recognized that a system containing multiple IDSs could have each IDS directed towards a different function. It would therefore have been obvious that the user devices described in Burns (nodes) could have host-based IDSs to detect and determine data patterns in a packet (*determining, by the user's device, at least one communication data pattern*). It would have been obvious for the network-based IDS to determine the *data type*, as discussed at [1.3] and [1.4].

- c. **[3.2] *modifying data packets corresponding to requests for sharing of responses corresponding to at least one data pattern, that are not compatible with the received privacy preference,***

202. As addressed at [17.3], [18.1], and [1.6], the administrator in Burns can set their desired configuration to to dictate which data packets are allowed based on their “*data type*” or “*data pattern.*” Burns, 5:30:34 (“a system administrator may

configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a specified list of identifiable applications, or prevent all communications except for communications from a specified list of permitted software applications.”)

203. As addressed in the analysis of [18.3], Burns’s IDS can be configured to take numerous actions in response to a packet that does not match the administrator’s configuration, such as (1) dropping the packet, (2) automatically closing the communication session, or (3) throttling the communication session. Burns, 7:47-51, 11:50-58. A POSITA would have recognized that each of (1) dropping the packet, (2) automatically closing the communication session, and (3) throttling the communication session involve “*modifying data packets.*”

204. A POSITA would have understood that an administrator’s configuration for the IDS to allow certain patterns would correspond to the “*privacy preference,*” rendering obvious “*modifying data packets corresponding to requests for sharing of responses corresponding to at least one data pattern, that are not compatible with the received privacy preference.*”

d. [3.3] *wherein the privacy preference comprises a list of allowed data patterns for sharing during communication.*

205. As addressed in the analysis of [17.3]-[17.4], [1.6], and [3.2], a POSITA would have recognized that the administrator’s configuration of desired *data patterns* to be identified by the IDS corresponds to the “*privacy preference.*”

As discussed at [17.8] and [18.1], Burns and Yang discuss examples of “*data patterns*:” (1) size, (2) encryption key exchanges, (3) repeated sequences or values.

As discussed at [18.1], Burns and Yang discuss attack definitions, patterns table, and anomalies table as examples of a “*data pattern database*.”

206. (1) Size: Burns describes that the administrator may configure the IDS to determine whether a packet’s size indicates that it is encrypted, and thus, which actions to take with a packet. Burns, 2:34-35 (“If the average size and frequency of data transmission matches known characteristics for a malicious or unwanted application, the IDS may block further communication of that session,”) 14:17-19 (“Another exemplary method for identifying encrypted packets includes tracking two classes for bytes of a packet and comparing the sizes of the classes,”), 14:42-43 (“Encryption detection module 58 may then determine the size N of the TCP/IP payload of the received packet.”) Therefore, it would have been obvious for the configuration information to include preferences relating to the size of a packet.

207. (2) Encryption key exchanges: Burns describes that the administrator can configure the IDS to determine how to treat the packet based on whether an encryption key was exchanged:

[W]here IDS 10 determines that a communication session, for which an **application cannot be identified**, is encrypted, IDS 10 may further determine whether a key exchange associated with the communication session can be identified; **where a key exchange has been identified**,

IDS 10 may permit the communication session and where a key exchange has not been identified, IDS 10 may terminate the communication session.

Burns, 5:64-6:4.

208. Therefore, a POSITA would have found it obvious for the configuration information to include preferences relating to whether a key exchange is observed.

209. (3) Repeated sequences or values: Burns describes that the administrator can configure the IDS to determine whether a packet is encrypted based on whether repeated sequences or values are present in the packet:

In one embodiment, encryption detection module 58 stores a sequence of data from each packet in a communication session and determines whether there exists **a repeated value in each packet of the communication session**; in this case, the repeated value may indicate a standard header of an unknown protocol.

Burns, 18:30-44.

210. Therefore, a POSITA would have found it obvious for the configuration information to include preferences relating to whether a repeated pattern indicates that an encrypted packet is being sent or received.

211. (4) Patterns table, attack definitions, and anomalies table: Burns describes that the stateful inspection engine of the IDS includes “patterns table 54, data buffer 55, anomalies table 56, and attack definitions 33.” Burns, 7:62-64. Yang provides further details about how anomaly and pattern information may be used: “stateful inspection engine 28 combines pattern matching with protocol-specific

anomaly analysis to detect sophisticated attack behaviors.” Yang, 11:49-51. Therefore, a POSITA would have found it obvious for the configuration information to include preferences relating to the patterns table, the attack definitions, and the anomalies table.

212. Burns describes that the IDS can be configured to block, automatically close, or throttle communications (“*modifying data packets*” per [3.2]) where the packets are not “*compatible*” with “*allowed data patterns*” per the administrator’s configuration (“*privacy preference*”), which renders claim 3 obvious.

4. Claim 4

- a. **[4.0]** *The method of claim 3, wherein the communication data pattern is determined based on a range selected from a group consisting of: data packet frequency, data packet size, data packet speed, data packet count, data packet ratio compared to other data type flow, data packet repetition, and data packet order.*

213. I understand from Cisco’s counsel that a recitation of each element listed after the colon is not necessary to be shown in the prior art. Nevertheless, my analysis below demonstrates multiple components listed in this claim.

214. As discussed at [18.1]-[18.2], Burns describes identifying a repeating pattern, sequence, or value across multiple packets:

In one embodiment, encryption detection module 58 stores a sequence of data from each packet in a communication session and determines **whether there exists a repeated value in each packet of the**

communication session; in this case, the repeated value may indicate a standard header of an unknown protocol.

Burns, 18:30-44.

215. A “repeated value” across multiple packets of a communication session renders obvious “*data packet repetition.*”

216. It would have been obvious that the “key exchange” in Burns would require for packets to be exchanged in a “*data packet order,*” as encryption keys must be exchanged in a specific order to complete the handshake and identify legitimate encrypted traffic. *See, e.g.* RFC 5246, 34; Kozierok, 162-65, 172-177.

217. A POSITA would have understood that an IDS could also look to “the average size and frequency of data transmission” to determine whether it “matches known characteristics for a malicious or unwanted application.” Burns, 2:33-36. Matching size or frequency of a transmission corresponds to “*data packet frequency*” and “*data packet size.*”

218. Therefore, the prior art renders obvious “*wherein the communication data pattern is determined based on a range selected from a group consisting of: data packet frequency, data packet size...data packet repetition, or data packet order.*”

5. Claim 5

- a. **[5.0] *The method of claim 1, wherein the communication data type is determined based on data packet characteristics selected from a group consisting of: data packet header, data packet footer, version number, IP (Internet Protocol)***

address, HTTPS (HyperText Transfer Protocol Secure), file extension, encryption method, encoding method, keywords, driver, and communication protocol.

219. I understand from Cisco's counsel that a recitation of each element listed after the colon is not necessary to be shown in the prior art. Nevertheless, my analysis below demonstrates multiple components listed in this claim.

220. As discussed at [1.3]-[1.4], both Burns and Yang determine the application or protocol (*data type*) by comparing a TCP port value to a list of known TCP port values that correspond to certain applications and protocols. *See* Burns, 4:36-42; Yang, 9:59-66, Table 1. A POSITA would have understood that a TCP port value corresponds to the claimed "*data packet characteristic*," because it defines how the packet is treated by the receiving device. For example, a packet with the TCP port value 80 corresponds to the HTTP protocol. Kozierok, 709. It would have been obvious that a packet on port 80 would be formatted according to HTTP protocol, thus, the protocol type corresponds to the to the "*data type*" of the packet. It would have also been obvious that the "*content*" of an HTTP packet would be web data.

221. The TCP port value (*data packet characteristic*) is a field of the TCP header (*data packet header*). Therefore, using the TCP port value to determine the protocol renders obvious "*the data type is determined based on data packet characteristics selected from a group consisting of: data packet header....*"

222. Just as the port number 80 would have been understood to correspond to a “*data packet characteristic*” indicating that the “*data type*” of the packet is the HTTP protocol, a POSITA would have recognized that other port numbers could be used to determine other applications (*data type[s]*). Kozierok, 709. For example, as described in Yang at Table 1, the port number 443 corresponds to HTTPS. Yang, Table 1. Therefore, using the TCP port value 443 to determine that the corresponding protocol is HTTPS renders obvious “*the communication data type is determined based on data packet characteristics selected from a group consisting of: ... HTTPS (HyperText Transfer Protocol Secure) ...*”

223. It would have been obvious to a POSITA that other fields of a packet header could be used to determine a “*data type*.” A POSITA would have recognized that “version information” is included in the packet header for an HTTP packet. Kozierok, 1346-48. As addressed at [3.2], the IDS of Burns and Yang may be configured to “prevent unwanted applications.” Burns, 4:1. Yang describes that the IDS may look for anomalies, such as missing HTTP version information, in the packet header to determine that the application is unwanted. Yang, 5:35-40 (“Other anomalies refer to protocol events...that may warrant a heightened level of scrutiny... Example anomalies for the HTTP protocol include missing HTTP version information...”). It would have been obvious that “version information” would have corresponded to a “*characteristic*” that would identify the application (*data type*),

and that “missing version information” would correspond to an unwanted application. Therefore, determining that a packet is of an unwanted type based on version information in the packet header renders obvious “*the communication data type is determined based on data packet characteristics selected from a group consisting of: ...version information.*”

224. A POSITA would have known that IP address is another field of a packet header. Kozierok, 330-33. As addressed at [1.6] and [3.2], the IDS of Burns and Yang can be configured to allow or prevent certain packets. Burns, 5:29-34. As addressed at [1.8], the IDS can “log information about the communication session associated with a packet” that has been dropped to keep track of unwanted packets. Burns, 11:51-52. Burns describes that “information” logged about a packet may include “source IP address” and “destination IP address.” Burns, 20:60-62. A POSITA would have recognized that an IP address corresponds to a “*data packet characteristic*” that is used to determine the “*data type*” of the corresponding application, rendering obvious “*wherein the communication data type is determined based on data packet characteristics selected from a group consisting of: ...IP (Internet Protocol) address.*”

225. Similarly, it would have been obvious that the protocol field of an IP header corresponds to a “*data packet characteristic*” that can be used to identify whether a packet’s protocol (*data type*) is TCP or UDP. *See* RFC791, 14

(“Protocol:... This field indicates the next level protocol used in the data portion of the internet datagram.”); RFC790, 6 (showing TCP assigned Internet Protocol Number 6); Kozierok, 457 (“in transport mode, the protocol protects the messaged passed down to IP from the transport layer...the appropriate headers are added in front of the transport (UDP or TCP) header. The IP header is then added in front of that by IP.”) Therefore, “*wherein the communication data type is determined based on data packet characteristics selected from a group consisting of:....*” By determining whether a packet corresponds to the UDP or TCP communication protocol, this renders obvious “*the data type is determined based on data packet characteristics selected from a group consisting of: ... communication protocol.*”

6. Claim 6

a. [6.0] *The method of claim 1, further comprising:*

226. Claim 6 is dependent on the “*method*” disclosed in claim 1, therefore [6.0] would have been obvious for the reasons addressed in the analysis of [1.0].

b. [6.1] *determining, by the remote server, a communication data pattern of the at least one data packet of the received request; and*

227. As discussed in the analysis of [1.0], Burns’s IDS corresponds to the “*remote server.*” The IDS contains the “stateful inspection engine,” which includes an encryption detection module and an attack detection module. Burns, Fig.2. Burns’s IDS can identify a pattern or other “repeating sequence[s]” in a packet.

Burns, 18:38. This allows the encryption detection module to identify whether a packet is encrypted, and allows the attack detection module to identify whether a packet is associated with a known network attack:

Another exemplary method for determining whether a packet is encrypted includes **identifying repeating patterns** of varying lengths, e.g., one byte, two bytes, four bytes, or other within a packet.

Burns, 14:32-35

In one embodiment, encryption detection module 58 compares data in the packet to other data in the packet to find a **repeating sequence**. In one embodiment, encryption detection module 58 stores a sequence of data from each packet in a communication session and determines whether there exists **a repeated value in each packet of the communication session**; in this case, the repeated value may indicate a standard header of an unknown protocol.

Burns, 18:30-44.

Each of attack definitions **33 specifies a combination of one or more patterns** specified within patterns table 54 and one or more protocol-specific anomalies specified within anomalies table 56.

Burns, 8:6-10.

228. Yang provides further detail about “protocol-specific anomaly analysis,” including using anomalies “to detect sophisticated attack **behaviors.**”

Yang, 11:49-51. Identifying whether a packet is encrypted or whether a packet’s behavior might be indicative of an attack corresponds to a “*determining... a communication data pattern.*”

229. Burns also describes analyzing packets to determine whether an encryption key has been exchanged:

In particular, when "Detect key exchange" check box 342A is selected, administrator user interface 300 sends a message to security management module 44 to detect key exchanges for monitored communication sessions, or to operate in a detect key exchange mode. Accordingly, security management module 44 instructs stateful inspection engine 28 to **analyze incoming packets to determine whether those packets represent a key exchange for their corresponding communication session**. Stateful inspection engine 28 then begins to determine whether incoming packets represent a key exchange as part of the communication session. In one embodiment, stateful inspection engine 28 may only determine whether a first set of packets of a communication session represents a key exchange.

Burns, 19:25-38.

230. Determining that packets represent a key exchange renders obvious *“determining...a communication data pattern of the at least one data packet of the received request.”*

- c. **[6.2] linking at least one response from the user's device to a type of data packet from the user's device,**

231. As discussed at [1.3], the combination of Burns and Yang render obvious determining the *data type* of a packet based on the TCP port number. Using the TCP port numbers, it would be possible to identify a protocol or application that corresponds to a *“type of data packet from the user's device.”*

232. When taken in context with [6.3], limitation [6.2] describes “*linking*” a “*response*” (a data packet) to a “*type of data packet... based on*” the packet’s “*communication data type.*” The ’824 patent does not use the term “*communication data type,*” but refers to a “*data type of a data packet.*” See, e.g. ’824 patent, 9:38-39, 8:26 (“the data type 207 of a data packet”). If “*type of data packet*” and “*communication data type*” indeed refer to the same aspect of a packet, limitations [6.2] and [6.3] appear to describe an iterative process where a packet’s “*type*” is both the input and the output of an identification process. Yang describes a similar process where a packet’s protocol (*type*) may be identified and then reclassified based on the response flow:

For example, HTTP and FTP packet flows may share similar characteristics.... [U]pon receiving a packet flow that, upon initial classification, may be either HTTP or FTP, application identification module 51 may **first select HTTP as the type of application** based on the higher priority given to the type of protocol. Attack detection module 52 may then apply one or more signatures to the packet flow, including HTTP-specific signatures, according to the HTTP protocol assumption. **Upon receiving a reply packet flow from the server,** application identification module 51 **may examine the reply packet flow and determine that the communication session shares more properties with an FTP communication session.** Application identification module 51 may then **re-classify the communication as an FTP communication session.**

Yang, 10:60-11:10.

233. Yang describes that the IDS may receive a packet identified as the protocol type HTTP, and can examine the reply packet to reclassify the communication as protocol type FTP. Therefore, the IDS of Burns and Yang renders obvious “*linking*” a “*response*” (a data packet) to a “*type of data packet... based on*” the packet’s “*communication data type*.”

234. Alternatively, if the terms “*type of data packet*” and “*communication data type*” in the ’824 patent refer to two different aspects of a packet, Burns and Yang render obvious linking a “*type of data packet*” to a “*communication data type*” through the application of an “attack definition,” which is used to identify “malicious” packets. Burns, 4:20-25; Yang, 11:20-23 (“To detect an attack or other malicious activity, attack detection module 52 applies attack definitions...”), 14:12-21 (“If the packet flow is malicious...stateful inspection engine 28 reacts accordingly, by, for example, discarding the packet flow...”). Burns and Yang describe packet flows which are unwanted or are not consistent with identified protocol as “malicious.” Burns, 3:2-3, 4:20-25; Yang, 8:10-12, 11:12-13. A POSITA would have appreciated that “malicious” data packets as identified in Burns and Yang would be “*a type of data packet*,” and that identifying a packet from the user’s device as malicious would have rendered obvious “*linking*” that packet to a “*type of data packet*.”

- d. [6.3] *wherein the linking is based on the communication data type and communication data pattern of the at least one response.*

235. As discussed at [6.2], Burns and Yang render obvious the claimed “linking” of a “response” to a “type of data packet” by identifying the protocol or application of a packet based on the previously identified protocol or application for the communication session. Yang, 10:60-11:10.

236. Furthermore, Yang describes that the identification of a protocol or application may be based on pattern information:

After identifying the beginning of the packet flow, application identification module 51 makes a preliminary determination of the type of application and protocol of the packet flow (81). This **preliminary determination may be based on the pattern of the received packet flow**, initial inspection of the payloads of the packets of the packet flow, the amount of data received in the packet flow or other characteristics. Yang, 12:17-23; *see also id.*, 13:25-26 (“pattern matching to determine the type of application and underlying protocol”).

Application identification module 51 may analyze the pattern of the TCP data reassembled from packet flow to make the initial determination. For example, application identification module 51 may inspect the packet flow to find characteristics of known applications and the corresponding protocols, such as HTTP, FTP sendmail, SMTP, etc.
Yang, 13:35-41.

237. Therefore, by identifying a protocol or application based on “the pattern of the received packet flow,” Burns and Yang render obvious identifying an application or protocol based on the “*communication data pattern*” of the initially identified application or protocol (*communication data type*).

238. Alternatively, Burns and Yang describe filtering traffic for “malicious” packets with “compound attack definitions.” Burns, 6:44-45; Yang, 12:55-67. Yang details that “compound attack definitions” can utilize “specified patterns” and “protocol anomalies,” which can be specified to occur “in a required order.” Yang, 12:55-67, 15:29-67 (“administrator specifies a protocol to which the compound attack definition pertains...the administrator may add patterns by selecting Add button 120...”).

239. Yang further describes that protocol-specific pattern matching may be used to identify whether a packet represents an attack:

For example, a system administrator may specify a compound network attack that includes the protocol anomaly of repeated FTP login failure and a pattern that matches a login username of “root.”

Yang, 5:55-59; *see also id.* 8:39-48 (identifying FTP as a protocol).

240. By identifying that a packet may be malicious based on the combination (*the linking*) of the packet’s protocol (*communication data type*) and the matched pattern (*communication data pattern*), Burns and Yang render obvious this limitation.

7. Claim 7

a. [7.0] *The method of claim 1, further comprising:*

241. Claim 7 is dependent on the “*method*” disclosed in claim 1, therefore [7.0] would have been obvious for the reasons addressed in the analysis of [1.0].

b. [7.1] *identifying a response corresponding to a request for sharing of data packets,*

242. As addressed at [1.1]-[1.2], the combination of Wittenberg and Burns discloses that user may be prompted with a web form, which corresponds to a “*request for retrieval.*” Burns, 10:39; Wittenberg, [0046]. A POSITA would have understood that a “*request for retrieval*” of a data packet from a user’s device would correspond to “*a request for sharing of data packets.*”

243. As addressed at [1.2], the user’s device can provide a response in the web form by filling out the fields as required. As addressed at [1.8] and [18.3], Burns’s IDS monitors network traffic bi-directionally, “*identify[ing] pairs of packet flows that collectively form a single communication session.*” Burns, 7:8-15. The IDS identifying both the incoming traffic and the response web form corresponds to “*identifying a response corresponding to a request for sharing of data packets.*”

c. [7.2] *wherein the identified response is not compatible with the received privacy preference; and*

244. As addressed at [1.6], Burns’s IDS functions according to an administrator’s desired configuration information. Burns, 1:21-35. The

administrator's configuration corresponds to a "*privacy preference*." As addressed at [1.7], the "*privacy preference*" may be a list of data types that are allowed or not allowed:

For example, a system administrator may configure IDS 10 to explicitly **allow all identifiable applications**, allow all applications except for a specified list of identifiable applications, or **prevent all communications except for communications from a specified list** of permitted software applications.

Burns, 5:29-34.

245. As addressed at [1.8] and [18.3], Burns's IDS may block, close, or throttle communications that do not match the administrator's configuration. The IDS described in Burns and Yang performs this function on all packets due to its bidirectional nature. Burns. 10:25-26 ("stateful inspection engine 28 may receive all packets of a particular communication session"); Yang 8:3-7 ("determine from the response from the server that the original packet flow was just an attempt to bypass IDS"), 8:18-22, 13:10-12 ("IDS 20 then waits for a response packet flow and reanalyzes the packet flow in light of the response").

246. A POSITA would have recognized that "*during communication*" refers to the whole of the communication session, encompassing both the request that is sent to the user's device, as well as a response that is sent back. Burns, 8:60.

Therefore, during communication, the “*identified response*” could have a data type that is not permitted to be shared based on the administrator’s configuration.

- d. **[7.3] *providing a response with at least one modified data packet corresponding to the request for sharing types of data packets.***

247. As addressed at [18.3] and [1.8]-[1.9], Burns discloses that the IDS can drop a packet, automatically close a communication session, or throttle a communication session, all of which correspond to “*modif[ying]*” a packet. Burns’s IDS is bi-directional and therefore can monitor traffic between a client and a server. Burns, 7:8-15 (“packet flows that collectively form a single communication session between a client and a server.”). Therefore, it would have been obvious for Burns’s IDS to analyze and subsequently modify packets containing a web form such as the login webpage described in Wittenberg. Wittenberg, Fig. 8. A POSITA would have recognized that the web form “*response*” could be modified to “*correspond to the request for sharing types of data packets*” in accordance with the “*privacy preferences.*”

8. Claim 8

- a. **[8.0] *The method of claim 7, wherein the modification of the at least one modified data packet is selected from the group consisting of data nullification, blocking, data randomization, content modification, change of encoding,***

***change of file template, change of header, change of footer,
addition of a predetermined data packet and encryption.***

248. I understand from Cisco’s counsel that a recitation of each element listed after the colon is not necessary to be shown in the prior art. Nevertheless, my analysis below demonstrates multiple components listed in this claim.

249. As discussed in the analysis of [18.3] and [1.8], Burns describes “dropping” packets based on the administrator’s configuration of the IDS. Burns, 7:47-51. Dropping packets corresponds to “*modification of the at least one modified data packet,*” as the ’824 patent explains that “blocking the packet” is one form of modification. ’824 patent, 2:44-50.

250. Similarly, Burns describes “automatically closing the communication session” as a potential action to take per the administrator’s configuration. Burns, 7:47-51, 10:6-9. To close a communication session would entail the modification of a packet header by adding a FIN flag as addressed above in [18.3] and [1.8]. Kozierok, 765. A POSITA would have understood TCP protocols and would have known that in order to close a TCP session, a packet is “*modified*.”

251. Burns describes that the IDS may “throttle down the communication session associated with the packet.” Burns, 11:53. A POSITA would have recognized that one way to throttle a session would be to adjust the TCP window size by modifying, or changing, the TCP header. Kozierok, 805-06.

252. Therefore, Burns renders obvious “*wherein the modification of the at least one modified data packet is selected from the group consisting of...blocking,...content modification,...change of header...*”

9. Claim 9

- a. **[9.0] *A method of dynamic management of private data during communication between a remote server and a user's device, the method comprising:***

253. Limitation [9.0] is substantially the same as [1.0] and would have been obvious for the reasons analyzed above.

- b. **[9.1] *receiving, by the user's device, a request for retrieval of at least one data packet from the user's device,***

254. Limitation [9.0] is substantially the same as [1.1] and would have been obvious for the reasons analyzed above.

- c. **[9.2] *wherein the user's device is configured to provide a response corresponding to the received request;***

255. Limitation [9.2] is substantially the same as [1.2] and would have been obvious for the reasons analyzed above.

- d. **[9.3] *determining, by the remote server, at least one communication data pattern corresponding to the received request,***

256. Limitation [9.3] is the same as [1.3], except “*communication data type*” of [1.3] is changed to “*communication data pattern.*”

257. As addressed in the analysis of [17.8], [18.1]-[18.3], and [3.1], Burns and Yang disclose multiple methods for determining “*allowed patterns of data*

packets” and “*communication data patterns.*” Burns describes that a pattern may consist of size, encryption key exchange, or repeated sequences or values. *See, e.g.* Burns, 19:30-33 (“instructs stateful inspection engine 28 to analyze incoming packets to determine whether those packets represent a key exchange.”), 14:32-35 (“encryption detection module 58 compares data in the packet to other data in the packet to find a repeating sequence.”).

258. Both Burns and Yang disclose that the identification of “*data patterns*” is conducted by aspects of the IDS (*remote server*). *See, e.g.* Burns, Figs. 2 and 3 (showing that the IDS contains the stateful inspection engine 28, and that the encryption detection module 58 is a component of the stateful detection engine.) Therefore, Burns renders obvious “*determining, by the remote server, at least one communication data pattern corresponding to the received request.*”

- e. **[9.4] wherein the at least one communication data pattern is determined in accordance with a behavior range of the communication data packet, and**

259. As described at [18.1]-[18.3], Burns and Yang disclose multiple methods for determining “*patterns of data packets.*” Burns describes that a pattern may consist of size, encryption key exchange, or repeated sequences or values. *See,* Burns, 19:30-33, 14:32-35.

260. Yang provides further detail about “protocol-specific anomaly analysis,” including using anomalies “to detect sophisticated attack **behaviors.**”

Yang, 11:49-51. Identifying whether a packet is encrypted or whether a packet's behavior might be indicative of an attack corresponds to a "*communication data pattern in accordance with a behavior range of the data packet.*"

261. Burns also describes analyzing packets to determine whether an encryption key has been exchanged. *See, e.g.* Burns, 19:25-38 ("security management module 44 instructs stateful inspection engine 28 to analyze incoming packets to determine whether those packets represent a key exchange for their corresponding communication session.")

262. A POSITA would have recognized that determining that packets represent a key exchange renders obvious determining a "*communication data pattern in accordance with a behavior range of the data packet.*"

- f.** **[9.5] *wherein the content of the at least one data packet is not read by the remote server for continued operation by the user's device in real time during communication between the remote server and the user's device;***

263. Limitation [9.5] is substantially identical to [17.10] and [1.5] and would have been obvious for the same reasons as analyzed above.

- g.** **[9.6] *receiving, by the user's device, a privacy preference for the user's device,***

264. Limitation [9.6] is substantially identical to [1.6] and would have been obvious for the same reasons as analyzed above.

h. [9.7] wherein the privacy preference comprises a list of allowed data patterns for sharing during communication;

265. Limitation [9.7] is substantially identical to [3.3]. As addressed in the analysis of [17.3]-[17.4], [1.6], and [3.2], a POSITA would have recognized that the administrator's configuration which specifies *data patterns* to be identified by the IDS, corresponds to the "*privacy preference.*"

i. [9.8] modifying data packets corresponding to requests for sharing of responses and corresponding data patterns that are not compatible with the received privacy preference; and

266. Limitation [9.8] is substantially similar to [3.2]. However, [3.2] recites "*corresponding to requests for sharing of responses corresponding to at least one data pattern,*" and [9.8] recites "*modifying data packets corresponding to requests for sharing of responses and corresponding data patterns.*" A POSITA would have understood that "*corresponding to at least one data pattern*" would also apply to "*corresponding data patterns,*" in the plural.

267. As addressed at [17.3] and [1.6], the administrator's configuration in Burns corresponds to a "*privacy preference,*" and is used to determine which data packets are allowed based on their "*data type*" or "*data pattern.*" Burns, 5:30:34. As addressed in the analysis of [18.3] and [3.1]-[3.3], Burns's IDS can be configured to take numerous actions in response to a packet that does not match the administrator's configuration, such as (1) dropping the packet, (2) automatically closing the

communication session, or (3) throttling the communication session. Burns, 7:47-51, 11:50-58. A POSITA would have recognized that each of (1) dropping the packet, (2) automatically closing the communication session, and (3) throttling the communication session involve “*modifying data packets.*”

- j. **[9.9] *maintaining communication between the remote server and the user's device, with sharing of the modified data packets.***

268. Limitation [9.9] is substantially identical to [1.9] and would have been obvious for the same reasons as analyzed above.

10. Claim 10

- a. **[10.0] *The method of claim 9, wherein the communication data pattern of the at least one response is determined from metadata of communication with the remote server.***

269. Claim 10 is similar to claim 2, but refers to a “*communication data pattern*” rather than a “*communication data type.*” As discussed at [9.3], Burns’s IDS corresponds to the “*remote server.*” The IDS may detect “*data patterns*” based on size, encryption key exchange, or repeated sequences or values. *See, e.g.* Burns, 2:34-35 (“If the average size and frequency of data transmission matches known characteristics for a malicious or unwanted application, the IDS may block further communication of that session,”), 19:30-33 (“instructs stateful inspection engine 28 to analyze incoming packets to determine whether those packets represent a key exchange.”), 18:41-43 (“a repeated value in each packet of the communication session; in this case, the repeated value may indicate a standard header of an

unknown protocol.”) Burns notes that this strategy requires a “long series of packet exchanges before detection of the unwanted application,” and may “fail to identify certain unwanted applications and may trigger false positives for desirable applications. Burns, 2:37-43. Nevertheless, Burns does not discourage the use of this method of identifying a transmission as malicious or unwanted.

270. A POSITA would have recognized that packet size is a form of “*metadata*.” See, e.g. U.S. Patent Publication No. 2013/0179753, [0078] (“The header 315 may include other metadata, which may include, but is not limited to: a packet type metadata, a packet size and/or length metadata, access control metadata, and so on.”); U.S. Patent Publication No. 2014/0207997, [0062] (“the header within a packet 360 may include packet metadata such as . . . the packet size, linkages to other packets, error correction checksums, etc.”)

271. A POSITA would have found it obvious that key exchange is a form of “*metadata*.” See, e.g. U.S. Patent Publication No. 2012/0303952, [0047] (“Usage constraints, such as whether the key can be used for signing, encryption, or key exchange may be part of the principal metadata.”)

272. A POSITA would have found it obvious that a repeated value in each packet of a communication session would establish a “*data pattern*.” As addressed at claim [2.0], the characteristic port value in a header (*metadata*) can be used to

determine a packet's *data type*. If an unknown value (*metadata*) is repeated in the header of multiple packets, this would be a *data pattern*.

11. Claim 11

a. [11.0] *The method of claim 9, further comprising:*

273. The “*method*” of [11.0] would have been obvious for the reasons discussed at [9.0].

b. [11.1] *determining, by the remote server, the communication data type of the at least one data packet of the received request; and*

274. There is no antecedent basis for the claimed “*communication data type*” in this limitation. However, determining the “*communication data type*” as recited in limitation [11.1] is substantially similar to [1.3], so [11.1] would have been obvious for the reasons discussed above.

c. [11.2] *modifying data packets corresponding to requests for sharing of responses and corresponding data patterns, corresponding to at least one communication data packet type, that are not compatible with the received privacy preference,*

275. Limitation [9.8] discloses “*modifying data packets corresponding to requests for sharing of responses and corresponding data patterns, that are not compatible with the received privacy preference.*” Limitation [11.2] adds an additional feature to [9.8], such that an identified “*data pattern*” also corresponds to “*at least one communication data packet type.*” This limitation is substantially identical to [1.8] and [1.3], therefore, the same analysis applies.

276. As addressed at [9.3], examples of *data patterns* in Burns are size, encryption key exchange, or repeated sequences or values. As addressed at [17.2], Burns and Yang disclose certain protocols and applications that correspond to *data types*. Burns describes that its IDS can be configured so “a system administrator may configure IDS 10 to explicitly allow all identifiable applications, allow all applications except for a **specified list of identifiable applications**, or prevent all communications except for communications from a **specified list of permitted software applications**.” Burns, 5:27-34. A POSITA would have realized that Burns’s IDS could detect both a *data pattern* and a *data type* for a packet, and the *data type* would not match the *privacy preference* which only allowed certain applications or protocols.

277. A POSITA would have realized that, in the instance a *data type* did not correspond to the privacy preferences, Burns’s IDS would have acted to *modify* the packet by blocking, automatically terminating the session, or throttling the packet. See Burns, 7:47-51, 11:50-58.

- d. [11.3] *wherein the privacy preference comprises a list of allowed data packet types for sharing during communication.*

278. Limitation [1.7] is substantially the same as [11.3]. A POSITA would have recognized that Burns discloses that certain applications (*data types*) may be explicitly allowed, whereas all others are prevented. Burns, 5:27-34. Burns’s

disclosure therefore renders obvious “*wherein the privacy preference comprises a list of allowed data packet types for sharing during communication.*”

12. Claim 12

- a. **[12.0] *The method of claim 11, wherein the communication data type is determined based on data packet characteristics selected from a group consisting of: data packet header, data packet footer, version number, IP (Internet Protocol) address, HTTPS (HyperText Transfer Protocol Secure), file extension, encryption method, encoding method, keywords, driver, and communication protocol.***

279. Limitation [12.0] is substantially the same as limitation [5.0] and would have been obvious for the reasons analyzed above.

13. Claim 13

- a. **[13.0] *The method of claim 9,***

280. Claim 13 is dependent on the “*method*” disclosed in claim 9, therefore [13.0] would have been obvious for the reasons addressed in the analysis of [9.0].

- b. **[13.1] *wherein the user's device is in communication with an external database with at least one communication data pattern corresponding to a request for sharing of data packets from the user's device, and***

281. As discussed at [17.3] and [18.1], the “security management module 44 receives configuration information from administrator 42,” and can store information in a “database, a text file, or any other appropriate structure.” Burns, 9:64-65, 20:65-67. Burns describes storing other “information that administrator 42 may find useful” relating to detected attacks in a database. Burns, 20:64-65.

Therefore, it would have been obvious to store the administrator's preferred configuration for *data patterns* in a "database."

282. Burns discloses that "[c]omputing nodes 8A-8N (computing nodes 8) represent any private computing device within enterprise network 5, including ... database servers." As each node is separated by the switch, a POSITA would have understood a node containing a "database" to be an "external database." See, Burns, Fig. 1.

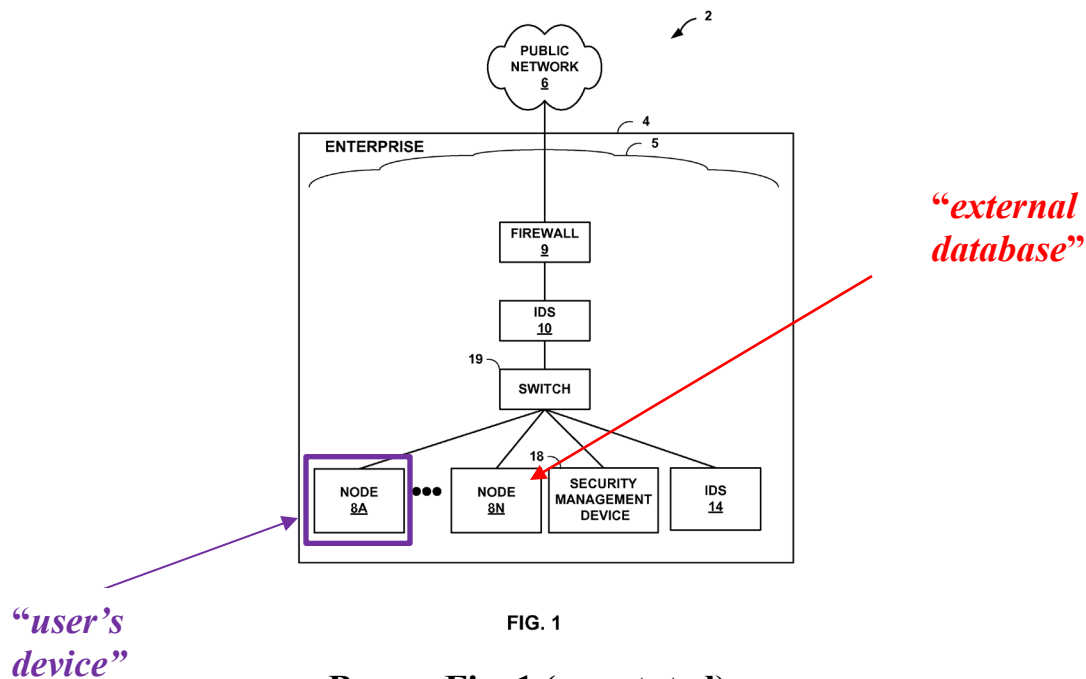


FIG. 1

Burns, Fig. 1 (annotated)

- c. [13.2] wherein at least one data pattern compatible with the received request is determined from the database.

283. As discussed at [9.6] and [9.7], the "privacy preference" is a list of the administrator's configuration for the IDS. The "privacy preference" list is used by

the IDS to determine if a *data pattern* is compatible with the received request. As discussed at [17.8] and [18.3], Burns and Yang describe identifying a *data pattern* corresponding to a packet. Because the pattern is allowed by the administrator, whether the pattern is “*compatible*” is determined by whether it corresponds to the administrator’s configuration (the *database*).

284. It would have been obvious that, as an alternative to the *privacy preference* being a “*list of allowed data patterns*,” that an “*external database*” would also contain “*at least one data pattern compatible with the received request.*”

14. Claim 14

a. [14.0] *The method of claim 9, further comprising:*

285. The method of claim [14.0] would have been obvious for the reasons discussed in [9.0].

b. [14.1] *determining, by the remote server, a type of the at least one data packet of the received request; and*

286. Limitation [14.1] is similar to limitation [1.3], which recites “*at least one communication data type of the at least one data packet corresponding to the received request.*” A POSITA would have understood that “*a type of the at least one data packet*” is substantially the same as the “*communication data type of the at least one data packet.*” Therefore, as per the analysis at [1.3], the prior art renders obvious “*determining, by the remote server, a type of the at least one data packet of the received request.*”

c. [14.2] *linking at least one data pattern to a type of data packet from the user's device,*

287. As addressed at [17.8], [1.3], and [1.4], an application or protocol as described in Burns and in Yang corresponds to a *data type*. As addressed at [17.8], [18.1], [18.2], and [3.1], Burns describes three examples of *data patterns*: size, encryption key exchange, or repeating values.

288. Burns describes that the IDS can determine if a *pattern* of data within a packet resembles a protocol (*data type*).

Protocol decoders 30 attempt to determine if the **packet resembles a protocol by, for example, looking for a particular pattern** of data at particular locations in the packet, such as in the portion of the TCP/IP payload that typically carries an application-layer header.

Burns, 10:65-11:2.

289. By determining that some patterns are associated with common protocols, Burns's IDS implicitly "*links*" the *data pattern* of a packet to the known protocol (*data type*). As addressed at [17.2], [1.3], and [1.8], Burns's IDS functions bi-directionally. Therefore, a POSITA would have recognized that Burns's IDS would be able to *link* a "*data pattern to a type of data packet from the user's device.*"

- d. **[14.3] wherein the linking is based on the communication data type and the communication data pattern of the at least one response.**

290. As addressed in [17.8] and [14.2], Burns's IDS would be able to "*link*" a *data pattern* and a *data type*. A POSITA would have recognized that Burns's IDS could first identify if a packet "resembles a particular protocol by, for example, looking for a particular pattern." Burns, 10:65-67. Burns's IDS could then confirm the *data type* by utilizing the TCP port value approach described in [1.3] and [1.4] to confirm the association (or *link*) between the identified *pattern* and the *data type*. Therefore, Burns and Yang render obvious this limitation.

15. Claim 15

- a. **[15.0] The method of claim 9, further comprising:**

291. The method of claim [15.0] would have been obvious for the reasons discussed in [9.0].

- b. **[15.1] identifying a communication data pattern corresponding to a request for sharing of data packet types,**

292. Limitation [15.1] is similar to [9.3], as it discloses determining the "*data pattern*" that corresponds to a "*request*" but adds the additional limitation that the request is for "*data packet types*" rather than a "*data packet*" as in [9.1]. As addressed at [19.0], Burns and Yang disclose "HTTP traffic and HTTP form data," which a POSITA would have recognized as web data and web form data. Burns, 10:39; Yang, 15:35-41. Wittenberg describes that "pop-up windows" may also

collect “additional information.” Wittenberg, [0046]. A POSITA would have recognized that the “additional information” might be a request to retrieve a protocol or application (*data packet type*) rather than a data packet, thus rendering obvious this limitation.

c. [15.2] *wherein the identified data pattern is not compatible with the received privacy preference; and*

293. As addressed in the analysis of [3.2] and of [9.3]-[9.7], Burns’s IDS (*remote server*) can determine the *data pattern* of the request and compare the *data pattern* to the administrator’s configuration information (*privacy preferences*). Burns discloses that certain data patterns do not correspond with the *privacy preferences*. Therefore, Burns renders obvious “*wherein the identified data pattern is not compatible with the received privacy preference.*”

d. [15.3] *providing a response with at least one modified data packet corresponding to the request for sharing types of data packets.*

294. As discussed at [18.3], [1.8]-[1.9], and [9.8], when a *data pattern* does not match the *privacy preference*, Burns’s IDS can respond by (1) dropping the packet, (2) automatically closing the communication session, or (3) throttling the communication session. Burns, 7:47-51, 11:50-58. Therefore, a POSITA would have recognized that Burns discloses “*providing a response with at least one modified data packet corresponding to the request for sharing types of data packets.*”

16. Claim 16

- a. **[16.0] *The method of claim 15, wherein the modification of the at least one modified data packet is selected from the group consisting of data nullification, blocking, data randomization, content modification, change of encoding, change of file template, change of header, change of footer, addition of a predetermined data packet and encryption.***

295. Claim limitation [16.0] is substantially the same as claim limitation [8.0], therefore, the same analysis applies.

XI. CONCLUSION

296. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true, and that these statements were made with knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under section 1001 of Title 18 of the United States Code.

Dated: 4/9/2025



Nader F. Mir, Ph.D.