

# **EXHIBIT 1**



US007647633B2

(12) **United States Patent**  
**Edery et al.**

(10) **Patent No.:** **US 7,647,633 B2**  
 (45) **Date of Patent:** **\*Jan. 12, 2010**

(54) **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**

(75) Inventors: **Yigal Mordechai Edery**, Pardesia (IL);  
**Nimrod Itzhak Vered**, Goosh Tai-Mond (IL);  
**David R. Kroll**, San Jose, CA (US);  
**Shlomo Touboul**, Kefar-Haim (IL)

(73) Assignee: **Finjan Software, Ltd.**, Netanya (IL)

(\* ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 917 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **11/159,455**

(22) Filed: **Jun. 22, 2005**

(65) **Prior Publication Data**

US 2006/0026677 A1 Feb. 2, 2006

**Related U.S. Application Data**

(63) Continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, and a continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, and a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**

**G06F 21/24** (2006.01)  
**G06F 11/30** (2006.01)  
**G06F 15/16** (2006.01)

(52) **U.S. Cl.** ..... **726/22**

(58) **Field of Classification Search** ..... None  
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,077,677 A 12/1991 Murphy et al. .... 706/62

(Continued)

FOREIGN PATENT DOCUMENTS

EP 1091276 4/2001  
 EP 1132796 9/2001

OTHER PUBLICATIONS

Zhong, et al., "Security in the Large: is Java's Sandbox Scalable?" *Seventh IEEE Symposium on Reliable Distributed Systems*, pp. 1-6, Oct., 1998.

(Continued)

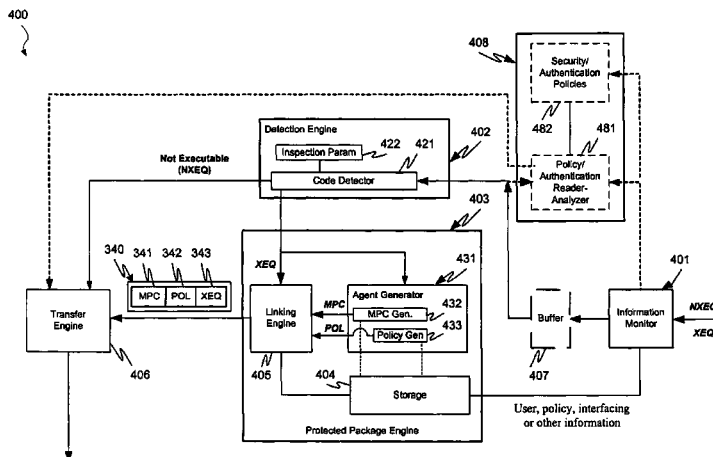
*Primary Examiner*—Christopher A Revak

(74) *Attorney, Agent, or Firm*—King & Spalding LLP

(57) **ABSTRACT**

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

**41 Claims, 10 Drawing Sheets**



## US 7,647,633 B2

Page 2

## U.S. PATENT DOCUMENTS

5,359,659	A	10/1994	Rosenthal	726/24
5,361,359	A	11/1994	Tajalli et al.	726/23
5,414,833	A	5/1995	Hershey et al.	726/22
5,485,409	A	1/1996	Gupta et al.	726/25
5,485,575	A	1/1996	Chess et al.	714/38
5,572,643	A	11/1996	Judson	709/218
5,579,509	A	11/1996	Furtney et al.	703/27
5,606,668	A *	2/1997	Shwed	726/13
5,623,600	A *	4/1997	Ji et al.	726/24
5,638,446	A	6/1997	Rubin	705/51
5,675,711	A	10/1997	Kephart et al.	706/12
5,692,047	A	11/1997	McManis	713/167
5,692,124	A	11/1997	Holden et al.	726/2
5,720,033	A	2/1998	Deo	726/2
5,724,425	A	3/1998	Chang et al.	705/52
5,740,248	A	4/1998	Fieres et al.	713/156
5,740,441	A	4/1998	Yellin et al.	717/134
5,761,421	A	6/1998	van Hoff et al.	709/223
5,765,205	A	6/1998	Breslau et al.	711/203
5,784,459	A	7/1998	Devarakonda et al.	713/165
5,796,952	A	8/1998	Davis et al.	709/224
5,805,829	A	9/1998	Cohen et al.	709/202
5,832,208	A	11/1998	Chen et al.	726/24
5,832,274	A	11/1998	Cutler et al.	717/171
5,850,559	A	12/1998	Angelo et al.	713/320
5,859,966	A	1/1999	Hayman et al.	726/23
5,864,683	A	1/1999	Boebert et al.	709/249
5,881,151	A	3/1999	Yamamoto	726/24
5,884,033	A	3/1999	Duvall et al.	709/206
5,892,904	A	4/1999	Atkinson et al.	726/22
5,951,698	A	9/1999	Chen et al.	714/38
5,956,481	A	9/1999	Walsh et al.	726/23
5,963,742	A	10/1999	Williams	717/143
5,974,549	A *	10/1999	Golan	726/23
5,978,484	A	11/1999	Apperson et al.	705/54
5,983,348	A *	11/1999	Ji	726/13
5,987,611	A	11/1999	Freund	726/4
6,088,801	A	7/2000	Grecsek	726/1
6,088,803	A	7/2000	Tso et al.	726/22
6,092,194	A *	7/2000	Touboul	726/24
6,154,844	A *	11/2000	Touboul et al.	726/24
6,167,520	A *	12/2000	Touboul	726/23
6,339,829	B1	1/2002	Beadle et al.	726/15
6,425,058	B1	7/2002	Arimilli et al.	711/134
6,434,668	B1	8/2002	Arimilli et al.	711/128
6,434,669	B1	8/2002	Arimilli et al.	711/128
6,480,962	B1 *	11/2002	Touboul	726/22
6,487,666	B1	11/2002	Shanklin et al.	726/23
6,519,679	B2	2/2003	Devireddy et al.	711/114
6,598,033	B2	7/2003	Ross et al.	706/46
6,732,179	B1 *	5/2004	Brown et al.	709/229
6,804,780	B1 *	10/2004	Touboul	713/181
6,917,953	B2	7/2005	Simon et al.	707/204
7,058,822	B2 *	6/2006	Edey et al.	726/22
7,210,041	B1	4/2007	Gryaznov et al.	713/188
7,343,604	B2	3/2008	Grabarnik et al.	719/313
7,418,731	B2	8/2008	Touboul	726/22
2004/0073811	A1	4/2004	Sanin	726/13
2004/0088425	A1	5/2004	Rubinstein et al.	709/230
2005/0172338	A1	8/2005	Sandu et al.	726/22
2006/0031207	A1	2/2006	Bjarnestam et al.	707/3

## OTHER PUBLICATIONS

Rubin, et al., "Mobile Code Security," *IEEE Internet*, pp. 30-34, Dec., 1998.

Schmid, et al. "Protecting Data From Malicious Software," *Proceeding of the 18<sup>th</sup> Annual Computer Security Applications Conference*, pp. 1-10, 2002.

Corradi, et al., "A Flexible Access Control Service for Java Mobile Code," *IEEE*, pp. 356-365, 2000.

International Search Report for Application No. PCT/IB97/01626, 3 pp., May 14, 1998 (mailing date).

International Search Report for Application No. PCT/IL05/00915, 4 pp., March 3, 2006.

Written Opinion for Application NO. PCT/IL05/00915, 5 pp., Mar. 3, 2006 (mailing date).

International Search Report for Application No. PCT/IB01/01138, 44 pp., Sep. 20, 2002 (mailing date).

International Preliminary Examination Report for Application No. PCT/IB01/01138, 2 pp., dated Dec. 19, 2002.

Gerzic, Amer, "Write Your Own Regular Expression Parser," Nov. 17, 2003, 18 pp..

Power, James, "Lexical Analysis," 4 pp., May 14, 2006, Retrieved from the Internet.

Sitaker, Kragen "Rapid Genetic Evolution of Regular Expression" [online], *The Mial Archive*, Apr. 24, 2004 (retrieved on Dec. 7, 2004), 5 pp..

"Lexical Analysis: DFA Minimization & Wrap Up" [online], Fall, 2004 [retrieved on Mar. 2, 2005], 8 pp..

"Minimization of DFA" [online], [retrieved on Dec. 7, 2004], 7 pp., Retrieved from the Internet:

"Algorithm: NFS -> DFA" [online], Copyright 1999-2001 [retrieved on Dec. 7, 2004], 4 pp..

"CS 3813: Introduction to Formal Languages and Automata - State Minimization and Other Algorithms for Finite Automata," [retrieved on May 11, 2003], 38 pp.

Watson, Bruce W., "Constructing Minimal Acyclic Deterministic Finite Automata," [retrieved on Mar. 20, 2005], 38 pp.

Change, Chia-Hsiang, "From Regular Expression to DFA's Using Compressed NFA's," Oct., 1992, 243 pp.

"Products," Articles published on the Internet, "Revolutionary Security for a New Computing Paradigm" regarding SurfinGate™, 7 pp. no date provided.

"Release Notes for the Microsoft, ActiveX Development Kit," Aug. 13, 1996, activex.adsp.or.jp/inetsdk/readme.txt, pp. 1-10.

Doyle et al., "Microsoft Press Computer Dictionary," Microsoft Press, 2d Edition, pp. 137-138, 1993.

Finjan Software Ltd., "Powerful PC Security for the New World of Java™ and Downloadables, SurfinShield™" Article published on the Internet by Finjan Software Ltd., 2 pp. 1996.

Finjan Software Ltd., "Finjan Announces as Personal Java™ Firewall for Web Browsers - The SurfinShield™ 1.6 (formerly known as SurfinBoard)," Press Release of Finjan Releases SurfinShield 1.6, pp., Oct. 21, 1996.

Finjan Software Ltd., "Finjan Announces Major Power Boost and New features for SurfinShield™ 2.0," Las Vegas Convention Center/Pavillion 5 P5551, 3 pp., Nov. 18, 1996.

Finjan Software Ltd., "Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product for the World Wide Web," Article published on the Internet by Finjan Software Ltd., Ip., Jul. 29, 1996.

Finjan Software Ltd., "Java Security: Issues & Solutions," Article published on the Internet by Finjan Software Ltd., 8 pp. 1996.

Finjan Software Ltd., Company Profile, "Finjan - Safe Surfing, the Java Security Solutions Provider," Article published on the Internet by Finjan Software Ltd., 3 pp., Oct. 31, 1996.

"IBM AntiVirus User's Guide, Version 2.4.," International Business Machines Corporation, pp. 6-7, Nov. 15, 1995.

Khare, R., "Microsoft Authenticode Analyzed" [online], Jul. 22, 1996 [retrieved on Jun. 25, 2003], 2 pp.

LaDue, M., Online Business Consultant: Java Security: Whose Business is It?, Article published on the Internet, Home Page Press, Inc., 4 pp., 1996.

Leach, Norvin, et al., "IE 3.0 Applets Will Earn Certification," *PC Week*, vol. 13, No. 29, 2 pp., Jul. 22, 1996.

Moritz, R., "Why We Shouldn't Fear Java," *Java Report*, pp. 51-56, Feb., 1997.

Microsoft, "Microsoft ActiveX Software Development Kit" [online], Aug. 12, 1996 [retrieved on Jun. 25, 2003], pp. 1-6.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet"

**US 7,647,633 B2**

Page 3

---

Microsoft Corporation, Oct., 1996, including Abstract, Contents, Introduction, and pp. 1-10.

Microsoft Corporation, Web Page Article "Frequently Asked Questions About Authenticode," last updated Feb. 17, 1997, printed Dec. 23, 1998.

Okamoto, E., et al., "ID-Based Authentication System for Computer Virus Detection," IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169-1170.

Omura, J. K., "Novel Applications of Cryptography in Digital Communications," *IEEE Communications Magazine*, pp. 21-29, May, 1990.

Schmitt, D.A., ".EXE files, OS-2 style," *PC Tech Journal*, vol. 6, No. 11, p. 76(13), Nov., 1988.

Zhang, X. N., "Secure Code Distribution," IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, pp. 76-79, Jun., 1997.

D. Grune, et al., "Parsing Techniques: A Practical Guide," John Wiley & Sons, Inc., New York, New York, USA, pp. 1-326, 2000.

\* cited by examiner

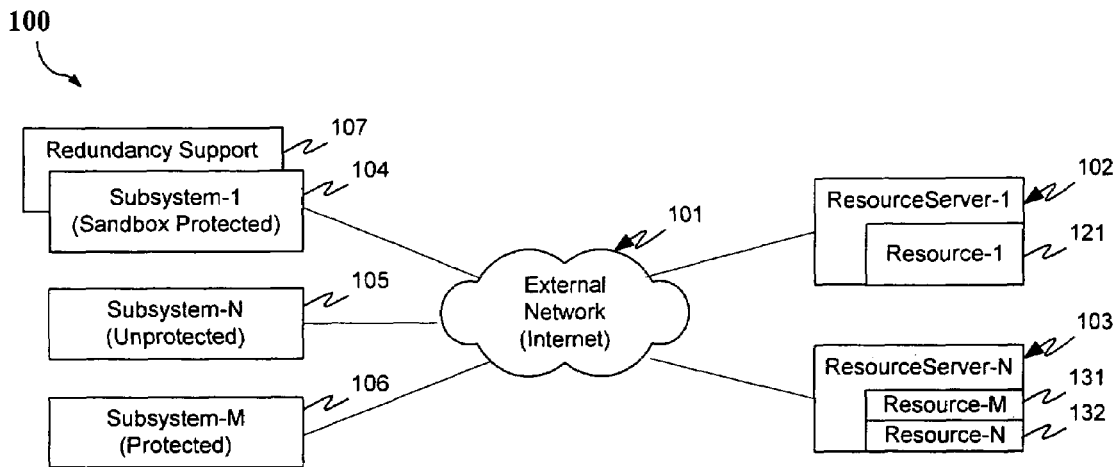


FIG. 1a

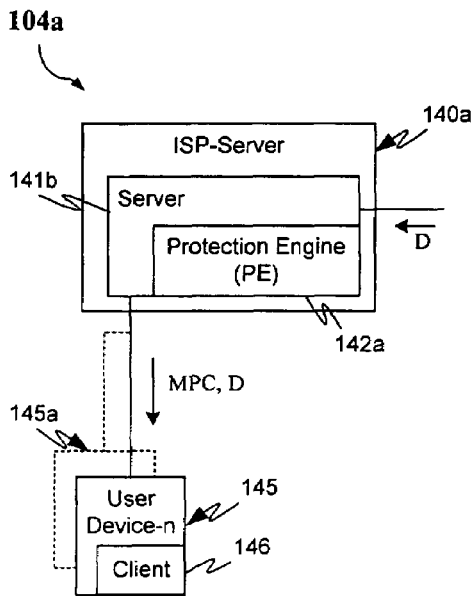


FIG. 1b

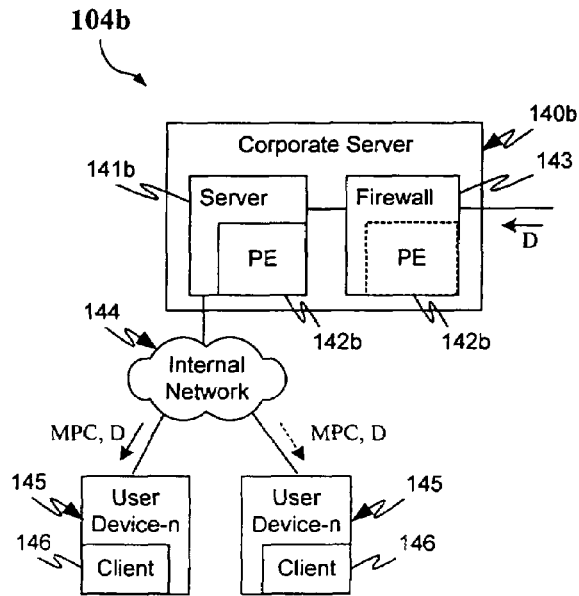


FIG. 1c

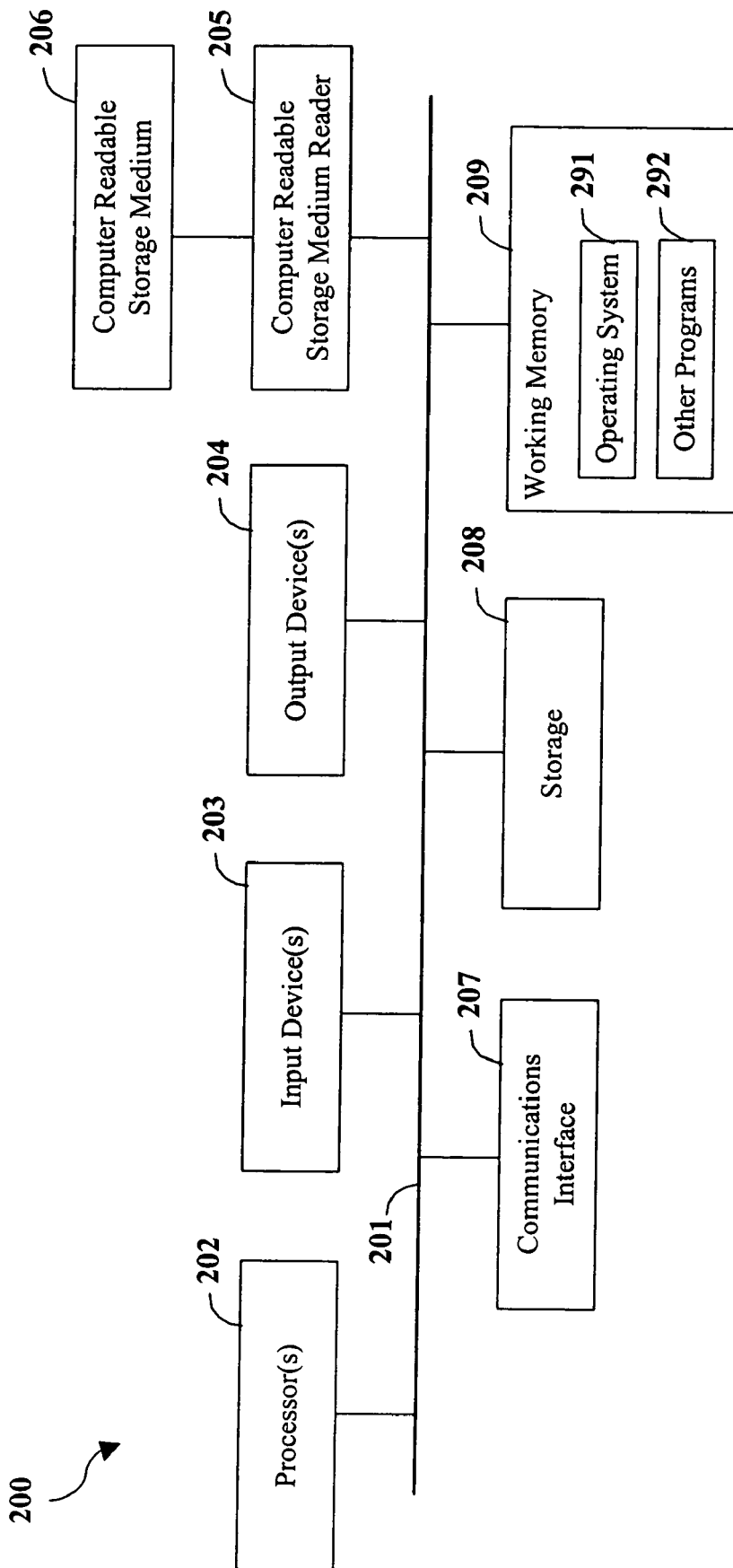


FIG. 2

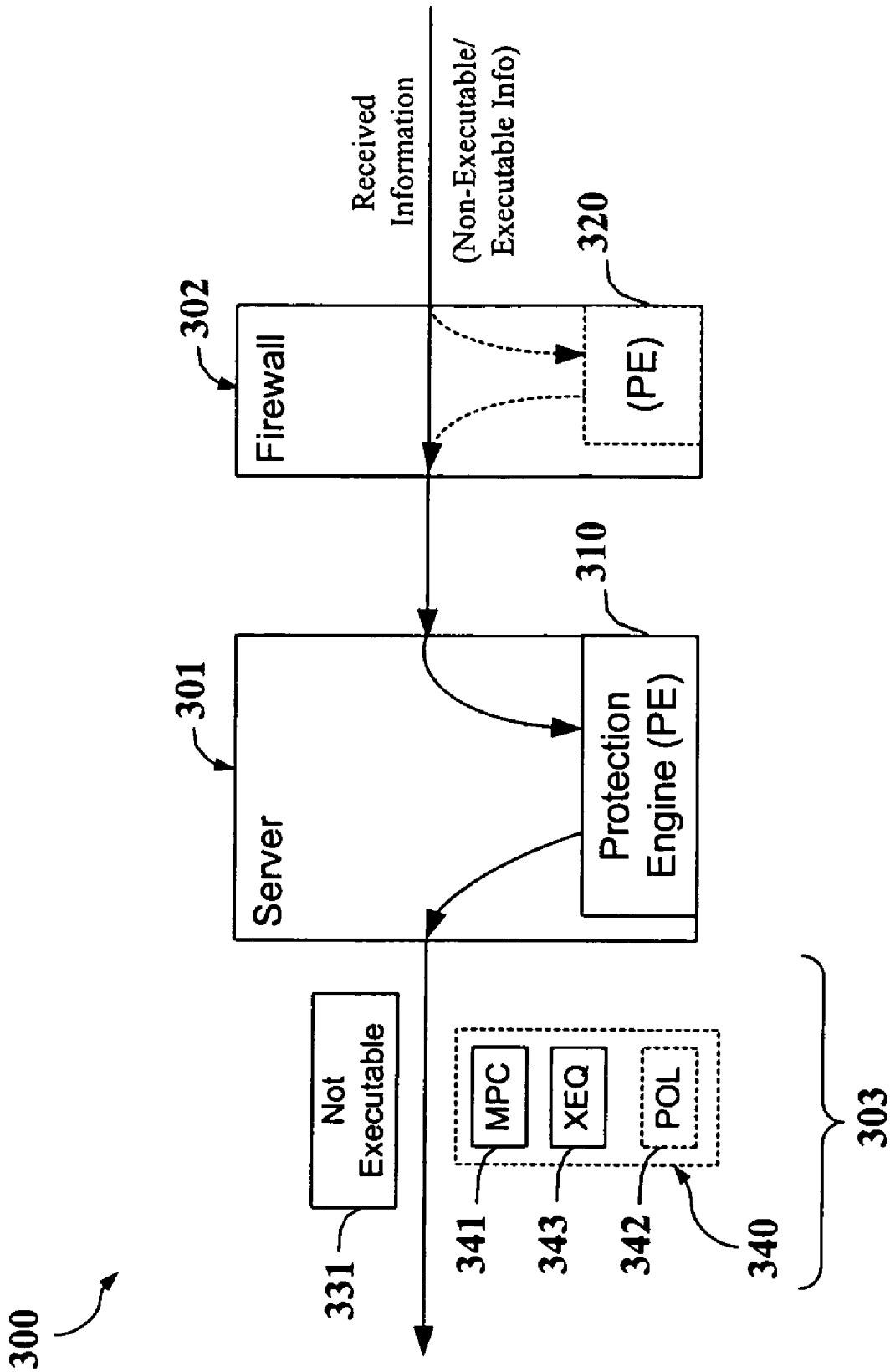


FIG. 3





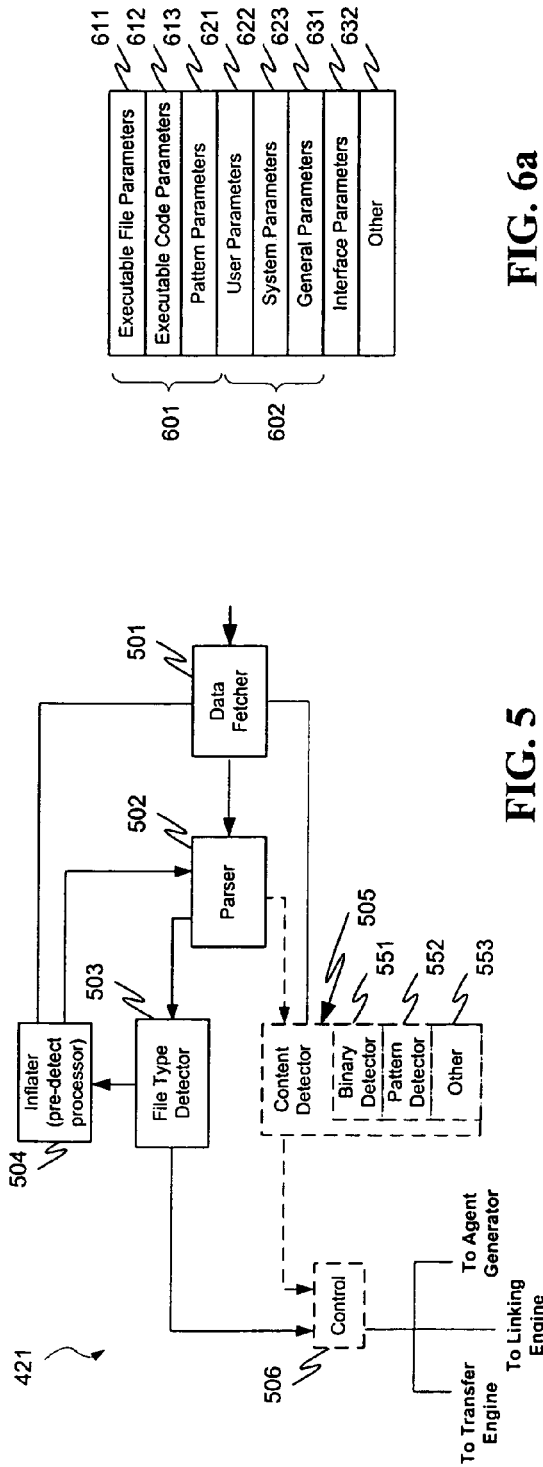


FIG. 5

FIG. 6a

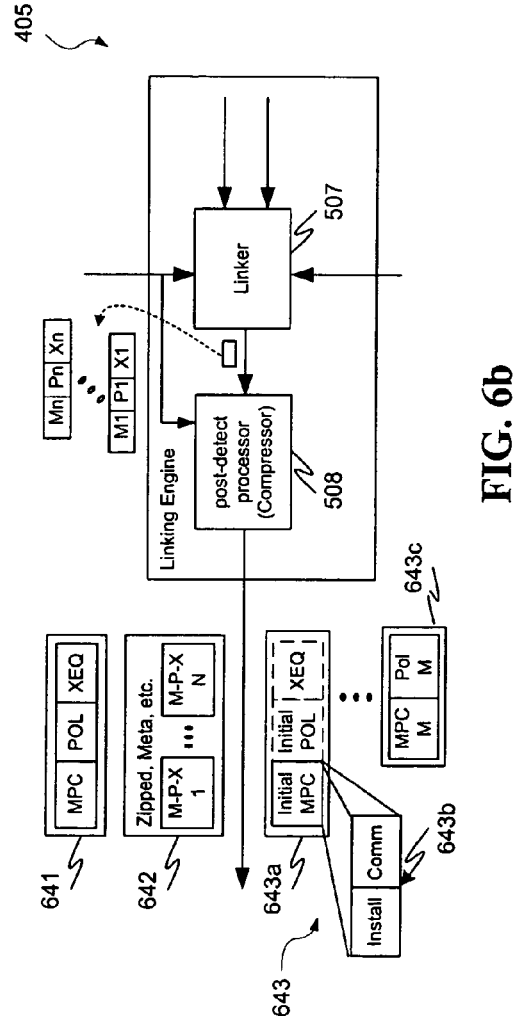
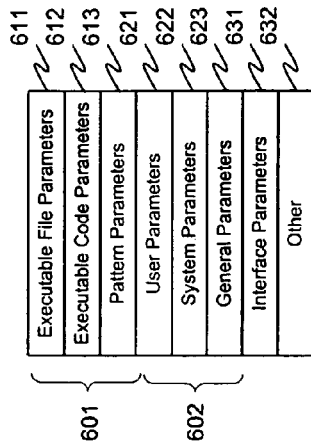


FIG. 6b

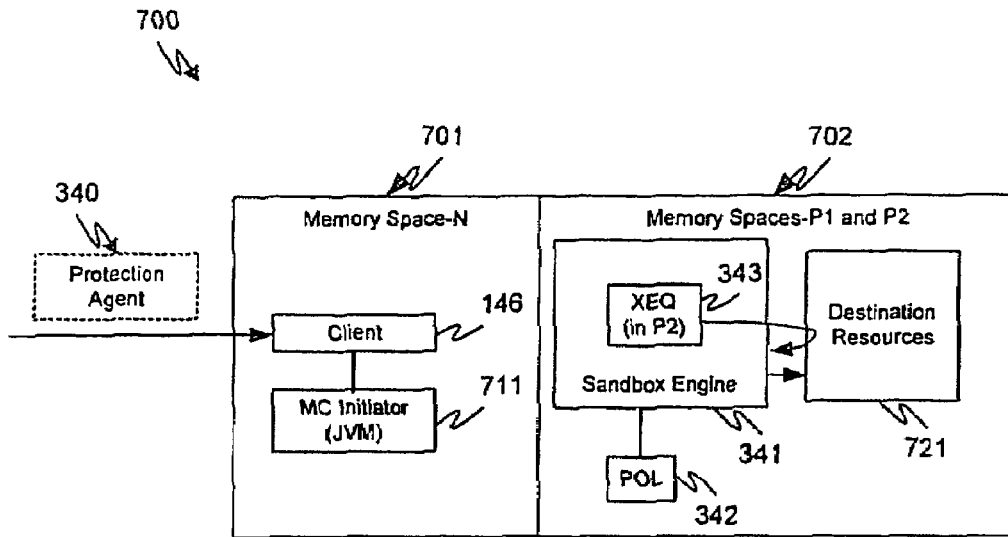


FIG. 7a

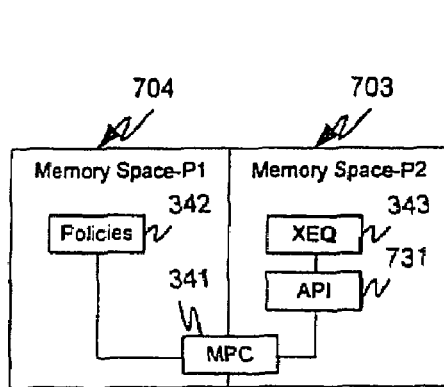


FIG. 7b

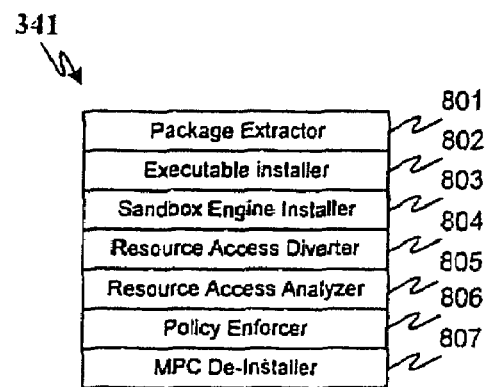


FIG. 8

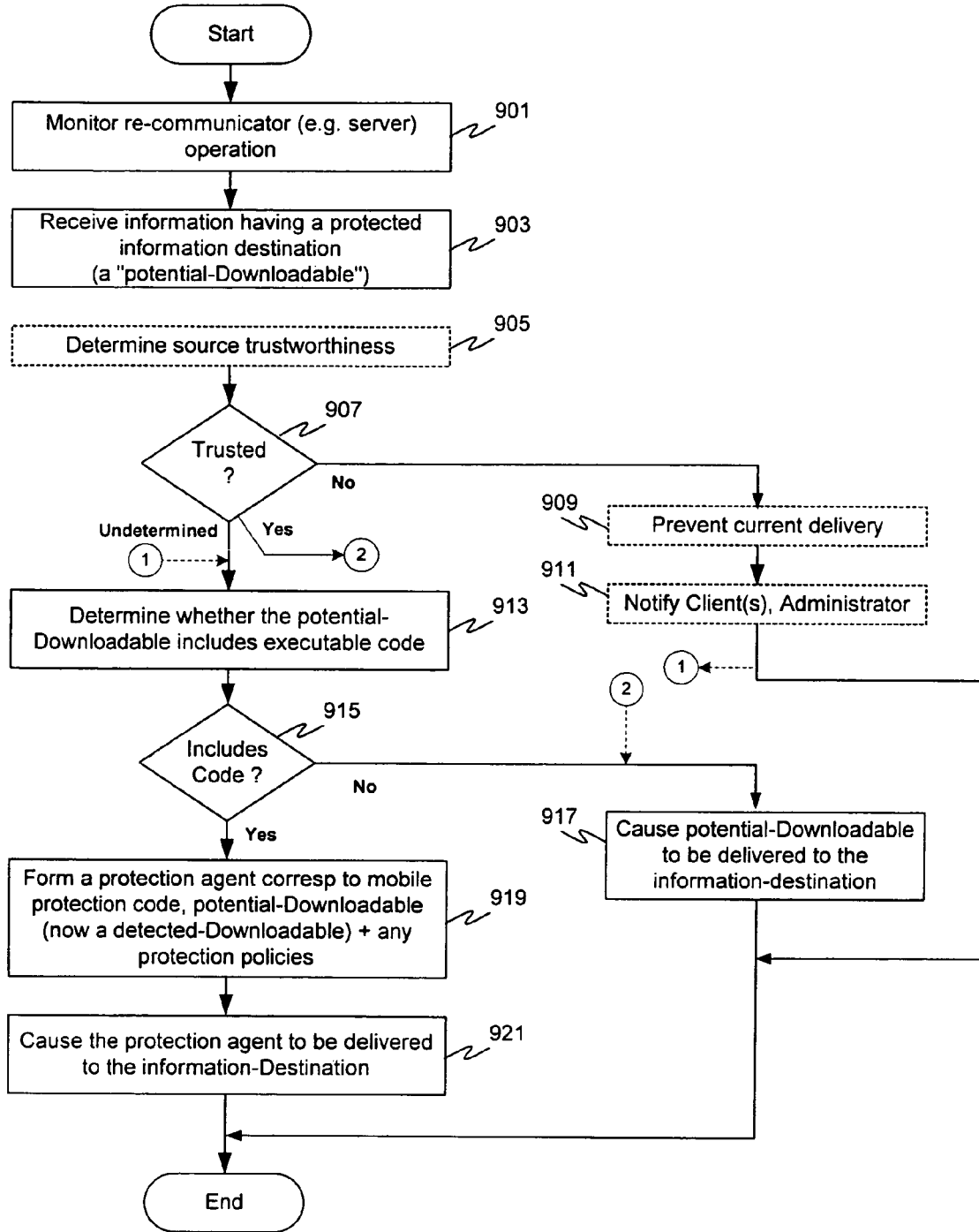


FIG. 9

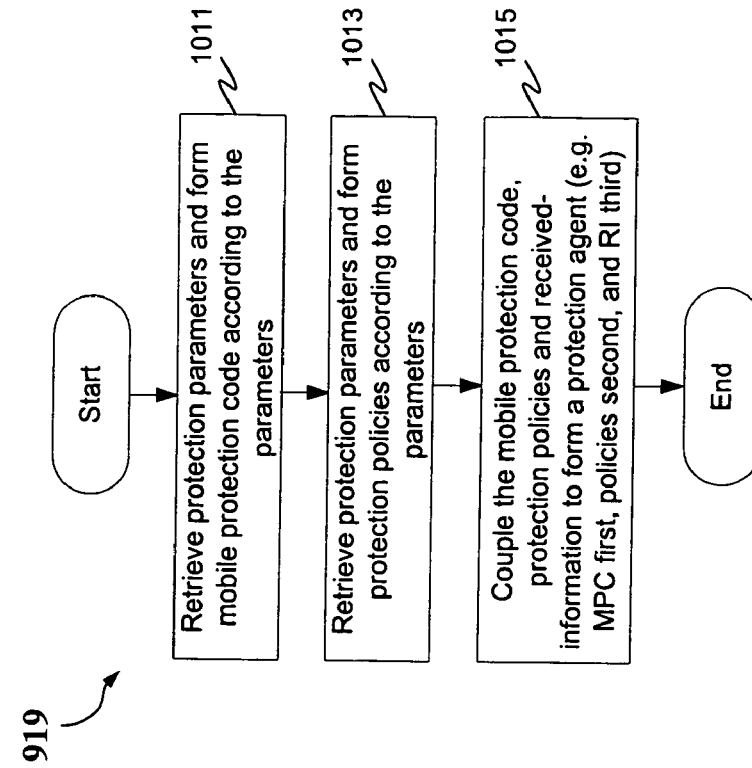


FIG. 10A

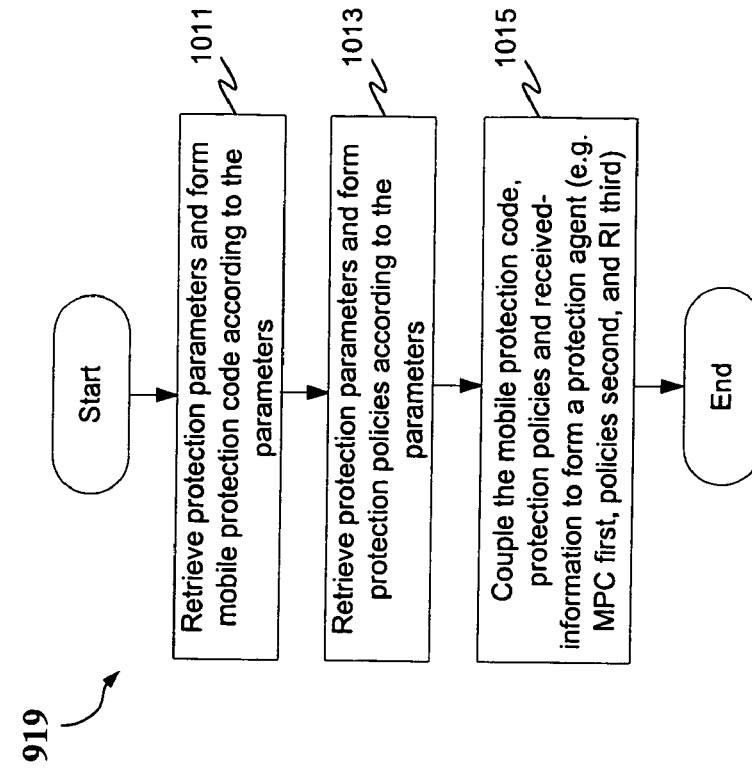


FIG. 10B

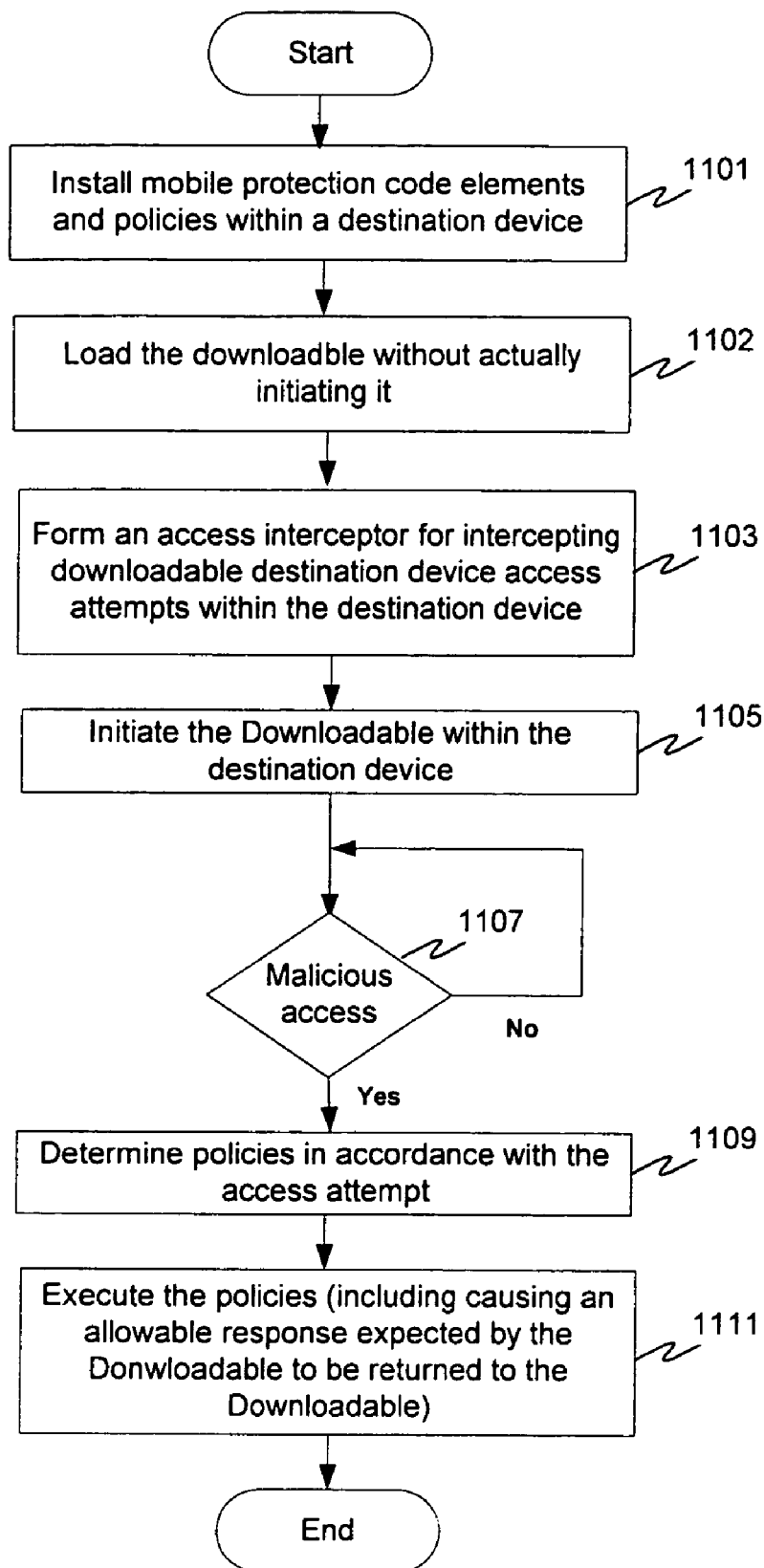


FIG. 11

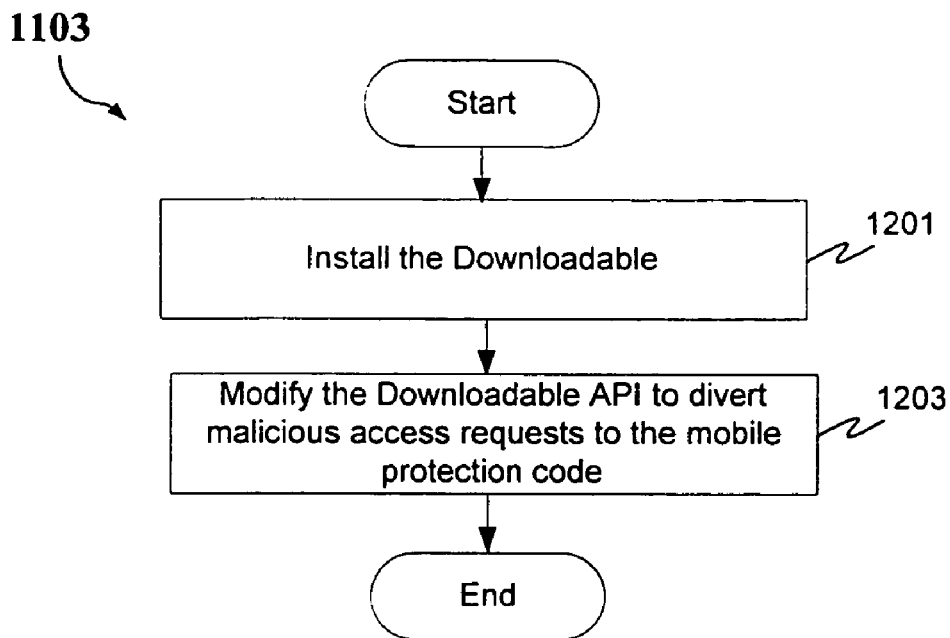


FIG. 12a

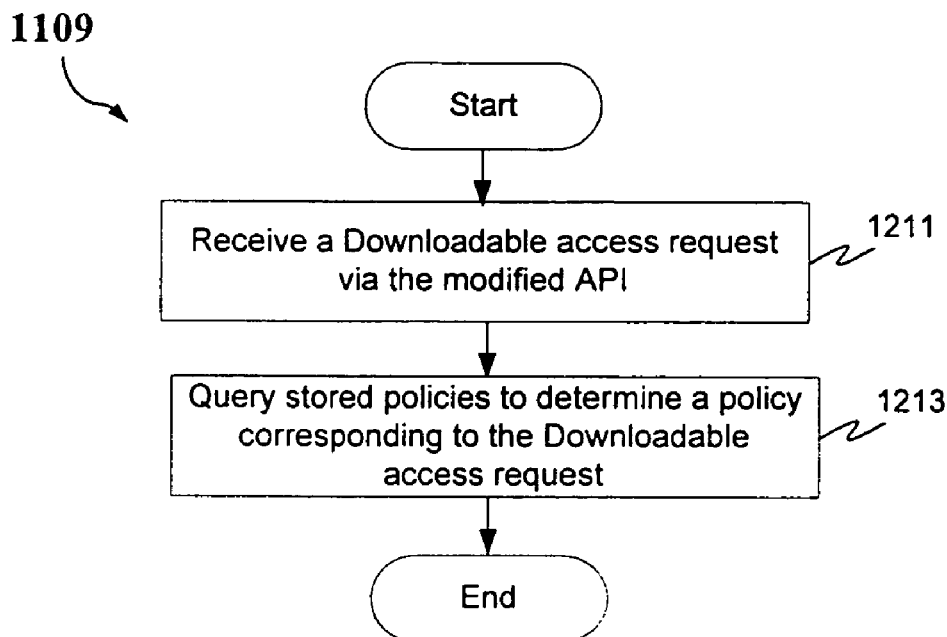


FIG. 12b

US 7,647,633 B2

1

**MALICIOUS MOBILE CODE RUNTIME  
MONITORING SYSTEM AND METHODS**PRIORITY REFERENCE TO RELATED  
APPLICATIONS

This application is a continuation of and incorporates by reference patent application Ser. No. 09/861,229, filed May 17, 2001 now U.S. Pat. No. 7,058,822, which claims benefit of reference provisional application Ser. No. 60/205,591 entitled “Computer Network Malicious Code Runtime Monitoring,” filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al. This application also incorporates by reference the provisional application Ser. No. 60/205,591. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled “System and Method for Protecting a Computer and a Network from Hostile Downloadables” filed on Mar. 30, 2000 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302, now U.S. Pat. No. 6,480,962, entitled “System and Method for Protecting a Client During Runtime From Hostile Downloadables”, filed on Apr. 18, 2000 by inventor Shlomo Touboul.

## BACKGROUND OF THE INVENTION

## 1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

## 2. Description of the Background Art

Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as “viruses.”

Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall or individually networked computer. The program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the new program operates successfully until yet another new virus appears—and so on. Of course, damage has already typically been incurred.

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only

2

distributable components including “Java applets or ActiveX controls”, and further does so using resource intensive and high bandwidth static Downloadable content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Pat. No. 5,974,549 to Golan teaches a protection system that further focuses only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. Pat. No. 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

## SUMMARY OF THE INVENTION

The present invention provides protection systems and methods capable of protecting a personal computer (“PC”) or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other “malicious” operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software “components”, such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

In one aspect, embodiments of the invention provide for determining, within one or more network “servers” (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a “Downloadable”). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-communicating devices (as are referred to herein summarily one or more “servers” or “re-communicators”). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically

US 7,647,633 B2

3

including mobile protection code and downloadable protection policies to be sent to a Downloadable-destination in conjunction with the received information, if the received information is determined to be a Downloadable.

A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code (“MPC”) for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective “sandbox”. The MPC/policies can further include a communicator for enabling further MPC/policy information or “modules” to be utilized and/or for event logging or other purposes.

A sandbox protection system according to an embodiment of the invention comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code (“MPC”), protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process (“user device”) for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base “module” and a “communicator” for enabling further up/downloading of one or more further “modules” or other information (e.g. events, user/user device information, etc.).

Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code (“MPC”) and protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The

4

method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by “hiding” undesirable operation causing information within apparently inert or otherwise “friendly” downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1a is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

FIG. 1b is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

FIG. 1c is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

FIG. 6a is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. 6b is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;



US 7,647,633 B2

5

FIG. 7a is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. 7b is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7a, according to an embodiment of the invention;

FIG. 8 is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

FIG. 9 is a flowchart illustrating a protection method according to an embodiment of the invention;

FIG. 10a is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

FIG. 10b is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

FIG. 12a is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. 12b is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

#### DETAILED DESCRIPTION

In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more “servers” (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both.

Embodiments further provide for causing mobile protection code (“MPC”) and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term “or”, as used herein, is generally intended to mean “and/or” unless otherwise indicated.) FIGS. 1a through 1c illustrate a computer network system 100 according to an embodiment of the invention. FIG. 1a broadly illustrates system 100, while FIGS. 1b and 1c illustrate exemplary protectable subsystem implementations corresponding with system 104 or 106 of FIG. 1a.

Beginning with FIG. 1a, computer network system 100 includes an external computer network 101, such as a Wide Area Network or “WAN” (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 102 and resource server-N 103). Where

6

external network 101 includes the Internet, resource servers 1-N (102, 103) might provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources 121, 131 and 132. Such information can also include more traditionally viewed “Downloadables” or “mobile code” (i.e. distributable components), as well as downloadable application programs or other further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

Also coupled via external network 101 are subsystems 104-106. Subsystems 104-106 can, for example, include one or more servers, personal computers (“PCs”), smart appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network 101 (e.g. using a dialup, DSL, cable modem, cellular connection, ERJRF, or various other suitable current or future connection alternatives). One or more of subsystems 104-106 might further operate as user devices that are connectable to external network 101 via an internet service provider (“ISP”) or local area network (“LAN”), such as a corporate intranet, or home, portable device or smart appliance network, among other examples.

FIG. 1a also broadly illustrates how embodiments of the invention are capable of selectively, modifiably or extensibly providing protection to one or more determinable ones of networked subsystems 104-106 or elements thereof (not shown) against potentially harmful or other undesirable (“malicious”) effects in conjunction with receiving downloadable information. “Protected” subsystem 104, for example, utilizes a protection in accordance with the teachings herein, while “unprotected” subsystem-N 105 employs no protection, and protected subsystem-M 106 might employ one or more protections including those according to the teachings herein, other protection, or some combination.

System 100 implementations are also capable of providing protection to redundant elements 107 of one or more of subsystems 104-106 that might be utilized, such as backups, failsafe elements, redundant networks, etc. Where included, such redundant elements are also similarly protectable in a separate, combined or coordinated manner using embodiments of the present invention either alone or in conjunction with other protection mechanisms. In such cases, protection can be similarly provided singly, as a composite of component operations or in a backup fashion. Care should, however, be exercised to avoid potential repeated protection engine execution corresponding to a single Downloadable; such “chaining” can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable.

FIGS. 1b and 1c further illustrate, by way of example, how protection systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a “client-server” configuration, as is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-server configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks, converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

US 7,647,633 B2

7

The FIG. 1*b* example shows how a suitable protected system 104*a* (which can correspond to subsystem-1 104 or subsystem-M 106 of FIG. 1) can include a protection-initiating host “server” or “re-communicator” (e.g. ISP server 140*a*), one or more user devices or “Downloadable-destinations” 145, and zero or more redundant elements (which elements are summarily depicted as redundant client device/process 145*a*). In this example, ISP server 140*a* includes one or more email, Internet or other servers 141*a*, or other devices or processes capable of transferring or otherwise “re-communicating” downloadable information to user devices 145. Server 141*a* further includes protection engine or “PE” 142*a*, which is capable of supplying mobile protection code (“MPC”) and protection policies for execution by client devices 145. One or more of user devices 145 can further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which MPC and protection policies are operable to protect user devices 145 from detrimental, undesirable or otherwise “malicious” operations of downloadable information also received by user device 145.

The FIG. 1*c* example shows how a further suitable protected system 104*b* can include, in addition to a “re-communicator”, such as server 142*b*, a firewall 143*c* (e.g. as is typically the case with a corporate intranet and many existing or proposed home/smart networks.) In such cases, a server 141*b* or firewall 143 can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server 141*b* and firewall 143, or in a more integrated-manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

System 104*b* also includes internal network 144 and user devices 145. User devices 145 further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which the MPCs or protection policies are operable. (As in the previous example, one or more of user devices 145 can also include or correspond with similarly protectable redundant system elements, which are not shown.)

It will be appreciated that the configurations of FIGS. 1*a*-1*c* are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacturer postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a “re-communicator” or “server”) for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or “user device” can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

FIG. 2 illustrates an exemplary computing system 200, that can comprise one or more of the elements of FIGS. 1*a* through 1*c*. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system 100 elements (FIGS. 1*a*-*c*) are implemented in hardware, soft-

8

ware or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

Computer system 200 comprises elements coupled via communication channels (e.g. bus 201) including one or more general or special purpose processors 202, such as a Pentium® or Power PC®, digital signal processor (“DSP”), etc. System 200 elements also include one or more input devices 203 (such as a mouse, keyboard, microphone, pen, etc.), and one or more output devices 204, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

System 200 also includes a computer readable storage media reader 205 coupled to a computer readable storage medium 206, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device 208 and memory 209, which can include hard disk variants, floppy/compact disk variants, digital versatile disk (“DVD”) variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices 207 can also be included, such as a modem, DSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but are not limited to those already discussed.

Working memory further includes operating system (“OS”) elements and other programs, such as application programs, mobile code, data, etc. for implementing system 100 elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or other aspects in accordance with a particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine (“JVM”), etc., and an application program interface (“API”), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

One or more system 200 elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system 200 element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device 207 or memory 208) in accordance with a particular application.

FIG. 3 illustrates an interconnected re-communicator 300 generally consistent with system 140*b* of FIG. 1, according to an embodiment of the invention. As with system 140*b*, system 300 includes a server 301, and can also include a firewall 302. In this implementation, however, either server 301 or firewall 302 (if a firewall is used) can further include a protection engine (310 or 320 respectively). Thus, for example, an included firewall can process received information in a con-

US 7,647,633 B2

9

ventional manner, the results of which can be further processed by protection engine **310** of server **301**, or information processed by protection engine **320** of an included firewall **302** can be processed in a conventional manner by server **301**. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

FIG. 3 also shows how information received by server **301** (or firewall **302**) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a “potential-Downloadable”, and received information found to include executable code will be referred to as a “Downloadable” or equivalently as a “detected-Downloadable” (regardless of whether the executable code includes one or more application programs, distributable “components” such as Java, ActiveX, add-in, etc.).

Protection engine **310** provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code (“MPC”) to be transferred to a device that is a destination of the potential-Downloadable (or “Downloadable-destination”). Protection engine **310** can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters alone or in combination (see below). Protection engine **310** can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g. which can also include protection mechanisms taught by the above-noted prior applications; see FIG. 4).

Operationally, protection engine **310** of server **301** monitors information received by server **301** and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine **310** further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine **310** enables the not executable potential-Downloadable **331** to be delivered to its destination in an unaffected manner.

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine **310** also causes mobile protection code or “MPC” **341** to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the detected-Downloadable **343** (see below). Protection engine **310** further causes downloadable protection policies **342** to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies **342** provide parameters (or can additionally or

10

alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

(One or more “checked”, tag, source, destination, type, detection or other security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine **310** by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

Further MPCs, protection policies or other information are also deliverable to a the same or another destination, for example, in accordance with communication by an MPC/ protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine **310** by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine **310** (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

(It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

System **300** enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

Providing an MPC upon each detection of a Downloadable also facilitates a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.) Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally downloadable “modules” upon server **301** detection of a Downloadable deliverable to the same user device, among other alternatives.

In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by caus-

US 7,647,633 B2

11

ing the MPC to be executed within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package **340** including MPC **341**, the detected-Downloadable **343** and any policies **342**. For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine **310** forms a protected package by concatenating, within sandboxed package **340**, MPC **341** for delivery to a Downloadable-destination first, followed by protection policies **342** and Downloadable **343**. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a JVM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package **340** by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package **340** as a single Downloadable. Such processing will, however, result in initiating the MPC **341** and—in accordance with further aspects of the invention—the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies **342**. (While system **300** is also capable of ascertaining protection policies stored at a Downloadable-destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

Turning to FIG. **4**, a protection engine **400** generally consistent with protection engine **310** (or **320**) of FIG. **3** is illustrated in accordance with an embodiment of the invention. Protection engine **400** comprises information monitor **401**, detection engine **402**, and protected packaging engine **403**, which further includes agent generator **431**, storage **404**, linking engine **405**, and transfer engine **406**. Protection engine **400** can also include a buffer **407**, for temporarily storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system **408**) Protection engine **400** can further provide for selectively re-directing, further directing, logging, etc. of a potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

(Note that FIG. **4**, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion, and should not be construed as exclusive or otherwise limiting.)

Information monitor **401** monitors potential-Downloadables received by a host server and provides the information via buffer **407** to detection engine **402** or to other system **400** elements. Information monitor **401** can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

Information monitor **401** can also provide for transferring, to storage **404** or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration information

12

(e.g. see FIG. **6**). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

Detection engine **402** includes code detector **421**, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters **422**, whether the potential-Downloadable includes executable code and is thus a “detected-Downloadable”. (Code detector **421** can also include detection processors for performing file decompression or other “decoding”, or such detection-facilitating processing as decryption, utilization/support of security system **408**, etc. in accordance with a particular application.)

Detection engine **402** further transfers a detected-downloadable (“XEQ”) to protected packaging engine **403** along with indicators of such detection, or a determined non-executable (“NXEQ”) to transfer engine **406**. (Inspection parameters **422** enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. **5**.) Detection engine **402** can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can further remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

Packaging engine **403** provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine **403** includes agent generator **431**, storage **404** and linking engine **405**.

Agent generator **431** includes an MPC generator **432** and a protection policy generator **433** for “generating” an MPC and a protection policy (or set of policies) respectively upon receiving one or more “generate MPC/policy” indicators from detection engine **402**, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator **432** and protection policy generator **433** provide for generating MPCs and protection policies respectively in accordance with parameters retrieved from storage **404**. Agent generator **431** is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy “modules” as initiated by further indicators such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms. (It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage **404**, but with a potential decrease in flexibility.)

MPC generator **432** and protection policy generator **433** are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of ser-

US 7,647,633 B2

13

viced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. 6). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable user devices/processes or other constraints.

Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX components, or other considerations). Agent generator 431 further transfers a resulting MPC and protection policy pair to linking engine 405.

Linking engine 405 provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine 405 is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in storage 404. Linking engine 405 can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing (e.g. see FIG. 6b).

It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine 405 is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package 340 as a concatenation of ordered elements including an MPC 341, applicable policies 342 and the Downloadable or "XEQ" 343 (e.g. see FIG. 4).

Linking engine 405 is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package 340 including one or more MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. 5 and 6)

Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destination resources can be minimized (via two or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in

14

accordance with the teachings herein. (It might, for example be desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

Policy/authentication reader-analyzer 481 summarily depicts other protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can further be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies 482). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be transferred with or without further processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

Finally, transfer engine 406 of protection agent engine 303 provides for receiving and causing linking engine 405 (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine 406 is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine 406 can also be provided in a more configurable manner, such as was already discussed for other system 400 elements. (Any one or more of system 400 elements might be configurably implemented in accordance with a particular application.) Transfer engine 406 can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. 5 with reference to FIG. 4, a code detector 421 example is illustrated in accordance with an embodiment of the invention. As shown, code detector 421 includes data fetcher 501, parser 502, file-type detector 503, inflater 504 and control 506; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher 501. (A coupling between data fetcher 501 and control 506 has been removed for clarity sake.)

Data fetcher 501 provides for retrieving a potential-Downloadable or portions thereof stored in buffer 407 or parameters from storage 404, and communicates such information or parameters to parser 502. Parser 502 receives a potential-Downloadable or portions thereof from data fetcher 501 and isolates potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

File type detector 502 receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader 502 can, for example, be configured to analyze a received potential-Downloadable for a file header, which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard ".exe" file format for Windows OS application programs, a Java class header for Java applets, and so on for other applications, distributed components, etc. "Zipped", meta or other compressed files, which might include one or more

US 7,647,633 B2

15

executables, also typically provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector **502** is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. “.exe” or “.com” in conjunction with a DOS or Microsoft Windows OS Downloadable-destination).

Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage **404** of FIG. **4**) which file type detector **502** can compare to a received potential-Downloadable. (Such delimiters can thus also be updated in storage **404** as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector **502** further transfers to controller **506** a detected file type indicator indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

In this example, the aforementioned detection processor is also included as pre-detection processor or, more particularly, a configurable file inflater **504**. File inflater **504** provides for opening or “inflating” compressed files in accordance with a compressed file type received from file type detector **503** and corresponding file opening parameters received from data fetcher **501**. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflater **504** returns such information to parser **502**. File inflater **504** also provides any now-accessible included executables to control **506** where one or more included files are to be separately packaged with an MPC or policies.

Control **506**, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control **506** sends the non-Downloadable to transfer engine **406** (FIG. **4**) along with any indicators that might apply. For an ordinary single-executable Downloadable, control **506** sends control information to agent generator **431** and the Downloadable to linking engine **405** along with any other applicable indicators (see **641** of FIG. **6b**). Control **506** similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable Downloadable, control **506** sends control information for each corresponding executable to agent generator agent generator **431**, and sends the executable to linking engine **405** along with controls and any applicable indicators, as in **643b** of FIG. **6b**. (The above assumes, however, that distributed downloading is not utilized; when used—according to applicable parameters—control **506** also operates in accordance with the following.)

Control **506** conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator **431**, linking engine **405** and transfer engine **406**. In the present example, control **506** initially sends controls to agent generator **431** and linking engine **405** (FIG. **4**) causing agent generator to generate an initial MPC and initial policies, and sends control and a detected-Downloadable to linking engine **405**. Linking engine **405** forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (**643a** of FIG. **6b**). An initial MPC within the sandboxed package includes an installer and a communicator and per-

16

forms installation as indicated below. The initial MPC also communicates via the communicator controls to control **506** (FIG. **5**) in response to which control **506** similarly causes generation of MPC-M and policy-M modules **643c**, which linking engine **405** links and transfer engine **406** causes to be sent to the Downloadable destination, and so on for any further such modules.

(It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control **506** or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

Regarding the remaining detection engine elements illustrated in FIG. **5**, where content analysis is utilized, parser **502** can also provide a Downloadable or portions thereof to content detector **505**. Content detector **505** can then provide one or more content analyses. Binary detector **551**, for example, performs detection of binary information; pattern detector **552** further analyzes the Downloadable for patterns indicating executable code, or other detectors can also be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control **506**. Alternatively, however, composite results from such analyses can also be sent to control **506** for evaluation. Control **506** can further conduct such evaluation in a summary manner (determining whether a Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.)

FIG. **6a** illustrates more specific examples of indicators/parameters and known (or “knowledge base”) elements that can be utilized to facilitate the above-discussed system **400** configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated “parameters.” It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

Executable parameters **601** comprise, in accordance with the above examples, executable file type parameters **611**, executable code parameters **612** and code pattern parameters **613** (including known executable file type indicators, header/code indicators and patterns respectively, where code patterns are utilized). Use parameters **602** further comprise user parameters **621**, system parameters **622** and general parameters **623** corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other protection, etc.). The remaining parameters include interface parameters **631** for providing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters **632**.

FIG. **6b** illustrates a linking engine **405** according to an embodiment of the invention. As already discussed, linking engine **405** includes a linker for combining MPCs, policies or

17

agents via concatenation or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine **405** also includes the aforementioned post-detection processor which, in this example, comprises a compressor **508**. As noted, compressor **508** receives linked elements from linker **507** and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine **508**.

FIGS. **7a**, **7b** and **8** illustrate a “sandbox protection” system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. **7a**, a client **146** receiving sandbox package **340** will “recognize” sandbox package **340** as a (mobile) executable and cause a mobile code installer **711** (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer **711** will also recognize sandbox package **340** as an executable and will attempt to initiate sandbox package **340** at its “beginning.” Protection engine **400** processing corresponding to destination **700** use of a such a loader, however, will have resulted in the “beginning” of sandbox package **340** as corresponding to the beginning of MPC **341**, as noted with regard to the above FIG. **4** example.

Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader **711**, for clarity sake) to initiate MPC **341**. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC **341** to effectively form a protection “sandbox” around Downloadable (e.g. detected-Downloadable or “XEQ”) **343**, to monitor Downloadable **343**, intercept determinable Downloadable **343** operation (such as attempted accesses of Downloadable **343** to destination resources) and, if “malicious”, to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly “safe” resource or resource operation, etc.).

MPC **341**, in the present OS example, executes MPC element installation and installs any policies, causing MPC **341** and protection policies **342** to be loaded into a first memory space, P1. MPC **341** then initiates loading of Downloadable **343**. Such Downloadable initiation causes OS loader **711** to load Downloadable **343** into a further working memory space-P2 **703** along with an API import table (“IAT”) **731** for providing Downloadable **631** with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9x and Windows NT), which can be accommodated by agent generator configurability, such as that given above. MPC **341** therefore has at least initial access to API IAT **731** of Downloadable **632**, and provides for diverting, evaluating and responding to attempts by Downloadable **632** to utilize system APIs **731**, or further in accordance with protection policies **342**. In addition to API diverting, MPC **341** can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in the MSDN or using other suitable methods.

18

Turning to FIG. **8** with reference to FIG. **7b**, an MPC **341** according to an embodiment of the invention includes a package extractor **801**, executable installer **802**, sandbox engine installer **803**, resource access diverter **804**, resource access (attempt) analyzer **805**, policy enforcer **806** and MPC de-installer **807**. Package extractor **801** is initiated upon initiation of MPC **341**, and extracts MPC **341** elements and protection policies **342**. Executable installer **802** further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes the operating system to initialize the Downloadable’s IAT **731** in the memory space of the downloadable process, P2, as already noted.

Sandbox engine installer **803** (running in process space P1) then installs the sandbox engine (**803-805**) and policies **342** into the downloadable process space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter **804** further modifies those Downloadable-API IAT entries that correspond with protection policies **342**, thereby causing corresponding Downloadable accesses via Downloadable-API IAT **731** to be diverted resource access analyzer **805**.

During Downloadable operation, resource access analyzer or “RAA” **805** receives and determines a response to diverted Downloadable (i.e. “malicious”) operations in accordance with corresponding protection policies of policies **342**. (RAA **805** or further elements, which are not shown, can further similarly provide for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit OS/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using “dummy” or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

The FIG. **9** flowchart illustrates a protection method according to an embodiment of the invention. In step **901**, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step **903**. Steps **905-911** depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be “unfriendly” and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any suitable alerts. (The protection engine might also continue to permit Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or avoid detection if the source is found to be “trusted”, among other alternatives enabled by the teachings herein.)

US 7,647,633 B2

19

If, in step **913**, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step **915**. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step **917**, and the method ends. If instead the potential-Downloadable is found to include executable code in step **915** (and is thus a “detected-Downloadable”), then the protection engine forms a sandboxed package in step **919** and causes the protection agent to be delivered to the information-Destination in step **921**, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

The FIG. **10a** flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step **1001**, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various combinations, etc.

Where content analysis is conducted, in step **1003** of FIG. **10a**, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step **1003**. Finally, in step **1005**, the protection engine determines whether the results of steps **1001** and **1003** indicate that the potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-Downloadable as a detected-Downloadable.

The FIG. **10b** flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step **1011**, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step **1013**, retrieves protection parameters and forms protection policies according to the parameters. Finally, in step **1015**, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example, include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

The FIG. **11** flowchart illustrates how a protection method performed by mobile protection code (“MPC”) according to an embodiment of the invention includes the MPC installing

20

MPC elements and policies within a destination device in step **1101**. In step **1102**, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended mode). The MPC further forms an access monitor or “interceptor” for monitoring or “intercepting” downloadable destination device access attempts within the destination device (according to the protection policies in step **1103**, and initiates a corresponding Downloadable within the destination device in step **1105**).

If, in step **1107**, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps **1109** and **1111**; otherwise the MPC returns to step **1107**. In step **1109**, the MPC determines protection policies in accordance with the access attempt by the Downloadable, and in step **1111**, the MPC executes the protection policies. (Protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

As shown in the FIG. **12a** example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable (but not executing it) in step **1201**. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **12b** example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable. As shown, the MPC receives the Downloadable access request via the modified IAT in step **1211**. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step **1213**.

The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

What is claimed is:

1. A computer processor-based method, comprising: receiving, by a computer, downloadable-information; determining, by the computer, whether the downloadable-information includes executable code; and based upon the determination, transmitting from the computer mobile protection code to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.
2. The method of claim 1, wherein the receiving includes monitoring received information of an information re-communicator.
3. The method of claim 2, wherein the information re-communicator is a network server.



US 7,647,633 B2

21

4. The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included type indicator indicating an executable file type.

5. The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included type detector indicating an archive file that contains at least one executable.

6. The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included file type indicator and an information pattern corresponding to one or more information patterns that tend to be included within executable code.

7. The method of claim 1, further comprising receiving, by the computer, one or more executable code characteristics of executable code that is capable of being executed by the information-destination, and wherein the determining is conducted in accordance with the executable code characteristics.

8. A computer processor-based system for computer security, the system comprising

an information monitor for receiving downloadable-information by a computer;

a content inspection engine communicatively coupled to the information monitor for determining, by the computer, whether the downloadable-information includes executable code; and

a protection agent engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated by the computer to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.

9. The system of claim 8, wherein the information monitor intercepts received information received by an information re-communicator.

10. The system of claim 9, wherein the information re-communicator is a network server.

11. The system of claim 8, wherein the content inspection engine comprises a file type detector for determining whether the downloadable-information includes a file type indicator indicating an executable file type.

12. The system of claim 8, wherein the content inspection engine comprises a parser for parsing the downloadable-information and a content analyzer communicatively coupled to the parser for determining whether one or more downloadable-information elements of the downloadable-information correspond with executable code elements.

13. A processor-based system for computer security, the system comprising:

means for receiving downloadable-information;

means for determining whether the downloadable-information includes executable code; and

means for causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.

14. A computer program product, comprising a computer usable medium having a computer readable program code therein, the computer readable program code adapted to be executed for computer security, the method comprising:

providing a system, wherein the system comprises distinct software modules, and wherein the distinct software modules comprise an information re-communicator and a mobile code executor;

receiving, at the information re-communicator, downloadable-information including executable code; and

22

causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.

15. The method of claim 14, wherein the mobile code executor is a Java Virtual Machine.

16. The method of claim 14, wherein the mobile code executor is the operating system, running native code executables.

17. The method of claim 14, wherein the mobile code executor is a subsystem of the operating system.

18. The method of claim 14, wherein the mobile code executor is a scripting host.

19. The method of claim 14, wherein the re-communicator is at least one of a firewall and a network server.

20. The method claim 14, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

21. A processor-based system for computer security, the system comprising:

receiving means for receiving, at an information re-communicator of a computer, downloadable-information, including executable code; and

mobile code means communicatively coupled to the receiving means for causing, by the computer, mobile protection code to be executed by a mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.

22. The system of claim 21, wherein the mobile code executor is a Java Virtual Machine.

23. The system of claim 21, wherein the mobile code executor is an operating system, running native code executables.

24. The system of claim 21, wherein the mobile code executor is a subsystem of the windows operating system.

25. The system of claim 21, wherein the mobile code executor is a scripting host.

26. The system of claim 21, wherein the re-communicator is at least one of a firewall and a network server.

27. The system of claim 21, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

28. A processor-based method, comprising:

receiving a sandboxed package that includes mobile protection code ("MPC") and a Downloadable and one or more protection policies at a computer at a Downloadable-destination;

causing, by the MPC on the computer, one or more operations attempted by the Downloadable to be received by the MPC;

receiving, by the MPC on the computer, an attempted operation of the Downloadable; and

initiating, by the MPC on the computer, a protection policy corresponding to the attempted operation.

29. The method of claim 28, wherein the sandboxed package is configured such that the MPC is executed first, the Downloadable is executed by the MPC and the protection policies are accessible to the MPC.

US 7,647,633 B2

23

30. The method of claim 28, wherein the causing comprises modifying, by the MPC, interfaces of a corresponding downloadable to resources at the destination.

31. The method of claim 30, wherein the modifying is accomplished by initiating a loading of the Downloadable, thereby causing a mobile code executor to provide and initialize the interfaces, modifying one or more interface elements to divert corresponding attempted Downloadable operations to the MPC, and initiating execution of the Downloadable.

32. The method of claim 30, wherein the interfaces comprise an import address table ("IAT") of a native code executable downloadable.

33. The method of claim 30, wherein modifying the interfaces installs a filter-driver between the downloadable and the resources.

34. A processor-based system for computer security, the system comprising:

- a mobile code executor on a computer for initiating received mobile code; and
- a sandboxed package capable of being received and initiated by the mobile code executor on the computer, the sandboxed package including a Downloadable and mobile protection code ("MPC") for causing one or more Downloadable operations to be intercepted by the computer and for processing the intercepted operations by the computer, if the Downloadable attempts to initiate the operations.

35. The system of claim 34, wherein the MPC comprises: an MPC installer for causing MPC elements to be installed; a Downloadable installer communicatively coupled to the MPC installer for installing the Downloadable; a resource access diverter communicatively coupled to the MPC installer for causing the Downloadable operations to be intercepted;

24

a resource access analyzer communicatively coupled to the MPC installer for receiving an intercepted Downloadable operation and determining a protection policy corresponding to the intercepted Downloadable operation; and

a policy enforcer communicatively coupled to the resource access analyzer for processing the intercepted Downloadable operation.

36. The system of claim 35, wherein the resource access diverter modifies one or more elements of an interface usable by the Downloadable to effectuate the Downloadable operations.

37. The system of claim 35, wherein the mobile code-executor is a Java Virtual Machine.

38. The system of claim 35, wherein the mobile code executor is an operating system, running native code executables.

39. The system of claim 35, wherein the mobile code executor is a subsystem of the operating system.

40. The system of claim 35, wherein the mobile code executor is a scripting host.

41. A processor-based system for computer security, the system comprising:

receiving means for receiving a sandboxed package that includes mobile protection code ("MPC") and a Downloadable and one or more protection policies at a Downloadable-destination;

monitoring means for causing, by the MPC, one or more operations attempted by the Downloadable to be received by the MPC;

second receiving means receiving, by the MPC, an attempted operation of the Downloadable; and initiating means for initiating, by the MPC, a protection policy corresponding to the attempted operation.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE

**CERTIFICATE OF CORRECTION**

PATENT NO. : 7,647,633 B2  
APPLICATION NO. : 11/159455  
DATED : January 12, 2010  
INVENTOR(S) : Edery et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

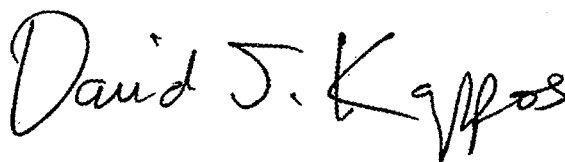
On the Title Page:

The first or sole Notice should read --

Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1043 days.

Signed and Sealed this

Twenty-eighth Day of December, 2010

A handwritten signature in black ink that reads "David J. Kappos". The signature is written in a cursive style with a large initial "D" and "K".

David J. Kappos

*Director of the United States Patent and Trademark Office*



US007647633C1

(12) **EX PARTE REEXAMINATION CERTIFICATE (10873rd)**  
**United States Patent**  
**Edery et al.** (10) **Number: US 7,647,633 C1**  
(45) **Certificate Issued: \*May 26, 2016**

(54) **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**

2221/033 (2013.01); G06F 2221/2119 (2013.01); G06F 2221/2141 (2013.01)

(75) Inventors: **Yigal Mordechai Edery**, Pardesia (IL); **Nimrod Itzhak Vered**, Goosh Tal-Moad (IL); **David R. Kroll**, San Jose, CA (US); **Shlomo Touboul**, Kefar-Haim (IL)

(58) **Field of Classification Search**  
None  
See application file for complete search history.

(73) Assignee: **FINJAN, INC.**, East Palo Alto, CA (US)

(56) **References Cited**

**Reexamination Request:**

No. 90/013,652, Dec. 9, 2015

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number 90/013,652, please refer to the USPTO's public Patent Application Information Retrieval (PAIR) system under the Display References tab.

**Reexamination Certificate for:**

Patent No.: **7,647,633**  
Issued: **Jan. 12, 2010**  
Appl. No.: **11/159,455**  
Filed: **Jun. 22, 2005**

Primary Examiner — Adam L Basehoar

(57) **ABSTRACT**

Certificate of Correction issued Dec. 28, 2010

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic Scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

(\*) Notice: This patent is subject to a terminal disclaimer.

**Related U.S. Application Data**

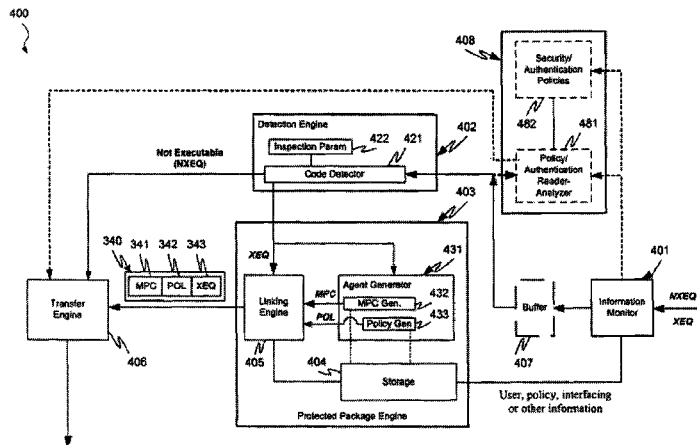
(63) Continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, and a continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, and a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**  
**G06F 11/30** (2006.01)  
**G06F 15/16** (2006.01)  
**G06F 21/53** (2013.01)  
**G06F 21/56** (2013.01)  
**H04L 29/06** (2006.01)  
**G06F 21/52** (2013.01)

(52) **U.S. Cl.**  
CPC ..... **G06F 21/53** (2013.01); **G06F 21/52** (2013.01); **G06F 21/562** (2013.01); **G06F 21/563** (2013.01); **H04L 63/145** (2013.01); **H04L 63/1408** (2013.01); **H04L 63/1441** (2013.01); **H04L 63/20** (2013.01); **G06F**

**At the time of issuance and publication of this certificate, the patent remains subject to pending reexamination control number 90/013,016 filed Oct. 7, 2013. The claim content of the patent may be subsequently revised if a reexamination certificate issues from the reexamination proceeding.**



US 7,647,633 C1

1

2

**EX PARTE  
REEXAMINATION CERTIFICATE**

NO AMENDMENTS HAVE BEEN MADE TO 5  
THE PATENT

AS A RESULT OF REEXAMINATION, IT HAS BEEN  
DETERMINED THAT:

The patentability of claims **8** and **12** is confirmed. 10  
Claims **1-7**, **9-11** and **13-41** were not reexamined.

\* \* \* \* \*



(12) **EX PARTE REEXAMINATION CERTIFICATE** (10939th)  
**United States Patent**  
**Edery et al.**

(10) **Number:** **US 7,647,633 C2**  
 (45) **Certificate Issued:** **\*Sep. 16, 2016**

(54) **MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS**  
 (75) **Inventors:** **Yigal Mordechai Edery**, Pardesia (IL); **Nimrod Itzhak Vered**, Goosh Tai-Mond (IL); **David R. Kroll**, San Jose, CA (US); **Shlomo Touboul**, Kefar-Haim (IL)

**H04L 29/06** (2006.01)  
**G06F 21/52** (2013.01)  
 (52) **U.S. Cl.**  
 CPC ..... **G06F 21/53** (2013.01); **G06F 21/52** (2013.01); **G06F 21/562** (2013.01); **G06F 21/563** (2013.01); **H04L 63/145** (2013.01); **H04L 63/1408** (2013.01); **H04L 63/1441** (2013.01); **H04L 63/20** (2013.01); **G06F 2221/033** (2013.01); **G06F 2221/2119** (2013.01); **G06F 2221/2141** (2013.01)

(73) **Assignee:** **FINJAN, INC.**, San Jose, CA (US)  
**Reexamination Request:**  
 No. 90/013,016, Oct. 7, 2013

(58) **Field of Classification Search**  
 None  
 See application file for complete search history.

**Reexamination Certificate for:**  
 Patent No.: **7,647,633**  
 Issued: **Jan. 12, 2010**  
 Appl. No.: **11/159,455**  
 Filed: **Jun. 22, 2005**

(56) **References Cited**

Reexamination Certificate C1 7,647,633 issued May 26, 2016

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number 90/013,016, please refer to the USPTO's public Patent Application Information Retrieval (PAIR) system under the Display References tab.

Certificate of Correction issued Dec. 28, 2010

(\*) **Notice:** This patent is subject to a terminal disclaimer.

*Primary Examiner* — Adam L Basehoar

**Related U.S. Application Data**

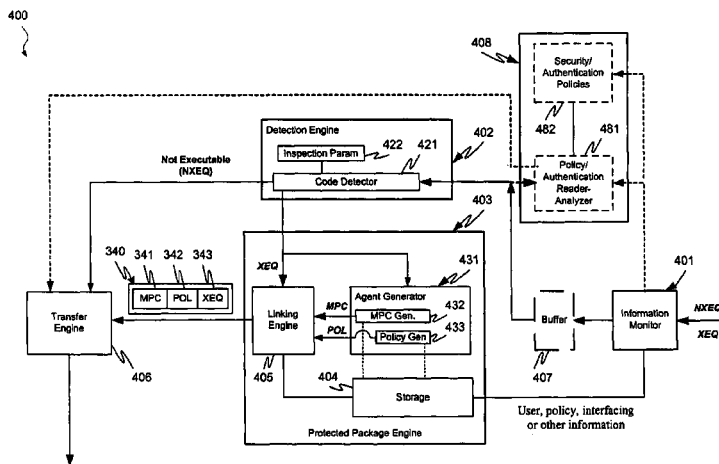
(57) **ABSTRACT**

(63) Continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, said application No. 11/159,455 is a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780, and a continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, which is a continuation of application No. 08/790,097, filed on Jan. 29, 1997, now Pat. No. 6,167,520, said application No. 09/539,667 is a continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Pat. No. 6,092,194.

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic Scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**  
**G06F 11/30** (2006.01)  
**G06F 15/16** (2006.01)  
**G06F 21/53** (2013.01)  
**G06F 21/56** (2013.01)



1

**EX PARTE  
REEXAMINATION CERTIFICATE**

THE PATENT IS HEREBY AMENDED AS  
INDICATED BELOW.

**Matter enclosed in heavy brackets [ ] appeared in the patent, but has been deleted and is no longer a part of the patent; matter printed in italics indicates additions made to the patent.**

ONLY THOSE PARAGRAPHS OF THE  
SPECIFICATION AFFECTED BY AMENDMENT  
ARE PRINTED HEREIN.

Column 1, lines 7-25:

This application is a continuation of and incorporates by reference patent application Ser. No. 09/861,229, filed May 17, 2001 now U.S. Pat. No. 7,058,822, which claims benefit of reference to provisional application Ser. No. 60/205,591 entitled "Computer Network Malicious Code Runtime Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al. This application also incorporates by reference the provisional application Ser. No. 60/205,591. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled "System and Method for Protecting a Computer and Network from Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul, which is a continuation of U.S. patent application Ser. No. 08/964,388, now U.S. Pat. No. 6,092,194, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables" filed on Nov. 6, 1997 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302 now U.S. Pat. No. 6,480,962, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul, which is a continuation of U.S. application Ser. No. 08/790,097, now U.S. Pat. No. 6,167,520 entitled "System and Method For Protecting a Client From Hostile Downloadables", filed Jan. 29, 1997 by inventor Shlomo Touboul.

AS A RESULT OF REEXAMINATION, IT HAS BEEN DETERMINED THAT:

The patentability of claims 1-7 and 28-33 is confirmed. New claims 42-45 are added and determined to be patentable. Claims 8-27 and 34-41 were not reexamined.

2

- 42. A computer processor-based method, comprising: receiving, by a computer, multiple instances of downloadable-information, wherein at least one of the multiple instances of downloadable-information includes non-executable information, at least one of the multiple instances of downloadable-information includes executable information and at least one of the multiple instances of downloadable-information includes a combination of non-executable and executable code portions; determining, by the computer, whether each of the multiple instances of downloadable-information includes executable code; and based upon the determination, transmitting from the computer mobile protection code to at least one information-destination of each instance of downloadable-information that is determined to include executable information and each instance of downloadable information that is determined to include a combination of non-executable and executable code portions.
- 43. A computer processor-based method, comprising: receiving, by a computer, downloadable-information; determining, by the computer, whether the downloadable-information includes executable code; and based upon the determination, transmitting from the computer mobile protection code and the downloadable-information to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code and transmitting the downloadable-information without the mobile protection code if the downloadable-information is determined not to include executable code.
- 44. A computer processor-based system for computer security, the system comprising: an information monitor for receiving downloadable-information by a computer; a content inspection engine communicatively coupled to the information monitor for determining, by the computer, whether the downloadable-information includes executable code, wherein determining if downloadable information includes executable code includes analyzing the downloadable information for operations to be executed on a computer; and a protection agent engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated by the computer to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.
- 45. The computer processor-based system of claim 48, wherein the content of the downloadable information is analyzed for one or more of binary information and a pattern indicative of executable code.

\* \* \* \* \*

# **EXHIBIT 2**





US007613926B2

(12) **United States Patent**  
**Edery et al.**

(10) **Patent No.:** **US 7,613,926 B2**  
 (45) **Date of Patent:** **\*Nov. 3, 2009**

(54) **METHOD AND SYSTEM FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

5,359,659 A 10/1994 Rosenthal ..... 726/24

(Continued)

(75) Inventors: **Yigal Mordechai Edery**, Pardesia (IL); **Nimrod Itzhak Vered**, Goosh Tel-Mond (IL); **David R. Kroll**, San Jose, CA (US); **Shlomo Touboul**, Kefar-Haim (IL)

FOREIGN PATENT DOCUMENTS

EP	1091276	4/2001
EP	1132796	9/2001

(73) Assignee: **Finjan Software, Ltd**, Netanya (IL)

(\* ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 659 days.

OTHER PUBLICATIONS

Zhong, et al., "Security in the Large: is Java's Sandbox Scalable?" *Seventh IEEE Symposium on Reliable Distributed Systems*, pp. 1-6, Oct. 1998.

This patent is subject to a terminal disclaimer.

(Continued)

(21) Appl. No.: **11/370,114**

(22) Filed: **Mar. 7, 2006**

*Primary Examiner*—Christopher A Revak

(65) **Prior Publication Data**

(74) *Attorney, Agent, or Firm*—King & Spalding LLP

US 2006/0149968 A1 Jul. 6, 2006

**Related U.S. Application Data**

(57) **ABSTRACT**

(63) Continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, and a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780, which is a continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Pat. No. 6,092,194, said application No. 09/861,229 is a continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962.

Protection systems and methods provide for protecting one or more personal computers ("PCs") and/or other intermittently or persistently network accessible devices or processes from undesirable or otherwise malicious operations of Java™ applets, ActiveX™ controls, JavaScript™ scripts, Visual Basic scripts, add-ins, downloaded/uploaded programs or other "Downloadables" or "mobile code" in whole or part. A protection engine embodiment provides, within a server, firewall or other suitable "re-communicator," for monitoring information received by the communicator, determining whether received information does or is likely to include executable code, and if so, causes mobile protection code (MPC) to be transferred to and rendered operable within a destination device of the received information, more suitably by forming a protection agent including the MPC, protection policies and a detected-Downloadable. An MPC embodiment further provides, within a Downloadable-destination, for initiating the Downloadable, enabling malicious Downloadable operation attempts to be received by the MPC, and causing (predetermined) corresponding operations to be executed in response to the attempts, more suitably in conjunction with protection policies.

(60) Provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**

**G06F 21/24** (2006.01)  
**G06F 11/30** (2006.01)  
**H04L 9/00** (2006.01)  
**G06F 15/16** (2006.01)

(52) **U.S. Cl.** ..... **713/181**; 713/175; 713/176; 726/24

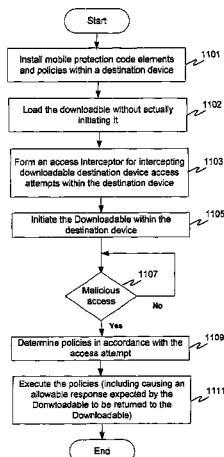
(58) **Field of Classification Search** ..... None  
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,077,677 A 12/1991 Murphy et al. .... 706/62

**30 Claims, 10 Drawing Sheets**



## US 7,613,926 B2

Page 2

## U.S. PATENT DOCUMENTS

5,361,359	A	11/1994	Tajalli et al.	726/23
5,414,833	A	5/1995	Hershey et al.	726/22
5,485,409	A	1/1996	Gupta et al.	726/25
5,485,575	A	1/1996	Chess et al.	714/38
5,572,643	A	11/1996	Judson	709/218
5,579,509	A	11/1996	Furtney et al.	703/27
5,606,668	A *	2/1997	Shwed	726/13
5,623,600	A *	4/1997	Ji et al.	726/24
5,638,446	A	6/1997	Rubin	705/51
5,675,711	A	10/1997	Kephart et al.	706/12
5,692,047	A	11/1997	McManis	713/167
5,692,124	A	11/1997	Holden et al.	726/2
5,720,033	A	2/1998	Deo	726/2
5,724,425	A	3/1998	Chang et al.	705/52
5,740,248	A	4/1998	Fieres et al.	713/156
5,740,441	A	4/1998	Yellin et al.	717/134
5,761,421	A	6/1998	van Hoff et al.	709/223
5,765,205	A	6/1998	Breslau et al.	711/203
5,784,459	A	7/1998	Devarakonda et al.	713/165
5,796,952	A	8/1998	Davis et al.	709/224
5,805,829	A	9/1998	Cohen et al.	709/202
5,832,208	A	11/1998	Chen et al.	726/24
5,832,274	A	11/1998	Cutler et al.	717/171
5,850,559	A	12/1998	Angelo et al.	713/320
5,859,966	A	1/1999	Hayman et al.	726/23
5,864,683	A	1/1999	Boebert et al.	709/249
5,881,151	A	3/1999	Yamamoto	726/24
5,884,033	A	3/1999	Duvall et al.	709/206
5,892,904	A	4/1999	Atkinson et al.	726/22
5,951,698	A	9/1999	Chen et al.	714/38
5,956,481	A	9/1999	Walsh et al.	726/23
5,963,742	A	10/1999	Williams	717/143
5,974,549	A	10/1999	Golan	
5,978,484	A	11/1999	Apperson et al.	705/54
5,983,348	A	11/1999	Ji	
5,987,611	A	11/1999	Freund	726/4
6,088,801	A	7/2000	Grecsek	726/1
6,088,803	A	7/2000	Tso et al.	726/22
6,092,194	A *	7/2000	Touboul	726/24
6,154,844	A *	11/2000	Touboul et al.	726/24
6,167,520	A	12/2000	Touboul	
6,339,829	B1	1/2002	Beadle et al.	726/15
6,425,058	B1	7/2002	Arimilli et al.	711/134
6,434,668	B1	8/2002	Arimilli et al.	711/128
6,434,669	B1	8/2002	Arimilli et al.	711/128
6,480,962	B1 *	11/2002	Touboul	726/22
6,487,666	B1	11/2002	Shanklin et al.	726/23
6,519,679	B2	2/2003	Devireddy et al.	711/114
6,598,033	B2	7/2003	Ross et al.	706/46
6,732,179	B1 *	5/2004	Brown et al.	709/229
6,804,780	B1 *	10/2004	Touboul	713/181
6,901,519	B1 *	5/2005	Stewart et al.	726/24
6,917,953	B2	7/2005	Simon et al.	707/204
7,058,822	B2 *	6/2006	Ederly et al.	726/22
7,093,135	B1 *	8/2006	Radatti et al.	713/188
7,210,041	B1	4/2007	Gryaznov et al.	713/188
7,343,604	B2	3/2008	Grabarnik et al.	719/313
7,418,731	B2	8/2008	Touboul	726/22
2004/0073811	A1	4/2004	Sanin	726/13
2004/0088425	A1	5/2004	Rubinstein et al.	709/230
2005/0172338	A1	8/2005	Sandu et al.	726/22
2006/0031207	A1	2/2006	Bjarnestam et al.	707/3

## OTHER PUBLICATIONS

Rubin, et al., "Mobile Code Security," *IEEE Internet*, pp. 30-34, Dec. 1998. Schmid, et al. "Protecting Data From Malicious Software," *Proceedings of the 18<sup>th</sup> Annual Computer Security Applications Conference*, pp. 1-10, 2002.

Corradi, et al., "A Flexible Access Control Service for Java Mobile Code," *IEEE*, pp. 356-365, 2000.

International Search Report for Application No. PCT/IB97/01626, 3 pp., May 14, 1998 (mailing date).

International Search Report for Application No. PCT/IL05/00915, 4 pp., dated Mar. 3, 2006.

Written Opinion for Application no. PCT/IL05/00915, 5 pp., dated Mar. 3, 2006 (mailing date).

International Search Report for Application No. PCT/IB01/01138, 4 pp., Sep. 20, 2002 (mailing date).

International Preliminary Examination Report for Application No. PCT/IB01/01138, 2 pp., dated Dec. 19, 2002.

Gerzic, Amer, "Write Your Own Regular Expression Parser," Nov. 17, 2003, 18 pp.

Power, James, "Lexical Analysis," 4 pp., May 14, 2006.

Sitaker, Kragen, "Rapid Genetic Evolution of Regular Expressions" [online], *The Mial Archive*, Apr. 24, 2004 (retrieved on Dec. 7, 2004), 5 pp.

"Lexical Analysis: DFA Minimization & Wrap Up" [online], Fall, 2004 [retrieved on Mar. 2, 2005], 8 pp.

"Minimization of DFA" [online], [retrieved on Dec. 7, 2004], 7 pp.

"Algorithm: NFS -> DFA" [online], Copyright 1999-2001 [retrieved on Dec. 7, 2004], 4 pp.

"CS 3813: Introduction to Formal Languages and Automata—State Minimization and Other Algorithms for Finite Automata," 3 pp., May 11, 2003.

Watson, Bruce W., "Constructing Minimal Acyclic Deterministic Finite Automata," [retrieved on Mar. 20, 2005], 38 pp.

Chang, Chia-Hsiang, "From Regular Expressions to DFA's Using Compressed NFA's," Oct. 1992, 243 pp.

"Products," Articles published on the Internet, "Revolutionary Security for a New Computing Paradigm" regarding SurfinGate™, 7 pp.

"Release Notes for the Microsoft ActiveX Development Kit," Aug. 13, 1996, pp. 1-10.

Doyle, et al., "Microsoft Press Computer Dictionary," Microsoft Press, 2d Edition, pp. 137-138, 1993.

Finjan Software Ltd., "Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™," Article published on the Internet by Finjan Software Ltd., 2 pp. 1996.

Finjan Software Ltd., "Finjan Announces a Personal Java™ Firewall for Web Browsers—the SurfinShield™ 1.6 (formerly known as SurfinBoard)," Press Release of Finjan Releases SurfinShield 1.6, 2 pp., Oct. 21, 1996.

Finjan Software Ltd., "Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0," Las Vegas Convention Center/Pavillion 5 P5551, 3 pp., Nov. 18, 1996.

Finjan Software Ltd., "Finjan Software Releases SurfinBoard, Industry's First JAVA Security Product for the World Wide Web," Article published on the Internet by Finjan Software Ltd., 1 p., Jul. 29, 1996.

Finjan Software Ltd., "Java Security: Issues & Solutions," Article published on the Internet by Finjan Software Ltd., 8 pp. 1996.

Finjan Software Ltd., Company Profile, "Finjan—Safe Surfing, The Java Security Solutions Provider," Article published on the Internet by Finjan Software Ltd., 3 pp., Oct. 31, 1996.

"IBM AntiVirus User's Guide, Version 2.4," International Business Machines Corporation, pp. 6-7, Nov. 15, 1995.

Khare, R., "Microsoft Authenticode Analyzed" [online], Jul. 22, 1996 [retrieved on Jun. 25, 2003], 2 pp.

LaDue, M., Online Business Consultant: Java Security: Whose Business is It?, Article published on the Internet, Home Page Press, Inc., 4 pp., 1996.

Leach, Norvin, et al., "IE 3.0 Applets Will Earn Certification," *PC Week*, vol. 13, No. 29, 2 pp., Jul. 22, 1996.

Moritz, R., "Why We Shouldn't Fear Java," *Java Report*, pp. 51-56, Feb. 1997.

Microsoft, "Microsoft ActiveX Software Development Kit" [Online], Aug. 12, 1996 [retrieved on Jun. 25, 2003], pp. 1-6.

Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet," Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction, and pp. 1-10.

Microsoft Corporation, Web Page Article "Frequently Asked Questions About Authenticode," last updated Feb. 17, 1997, printed Dec. 23, 1998, pp. 1-13.

**US 7,613,926 B2**

Page 3

---

Okamoto, E., et al., "ID-Based Authentication System for Computer Virus Detection," *IEEE/IEE Electronic Library online, Electronics Letters*, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169-1170.

Omura, J. K., "Novel Applications of Cryptography in Digital Communications," *IEEE Communications Magazine*, pp. 21-29, May 1990.

Schmitt, D.A., ".EXE files, OS-2 style," *PC Tech Journal*, vol. 6, No. 11, p. 76(13), Nov. 1988.

Zhang, X. N., "Secure Code Distribution," *IEEE/IEE Electronic Library online, Computer*, vol. 30, Issue 6, pp. 76-79, Jun. 1997.

D. Grune, et al., "Parsing Techniques: A Practical Guide," John Wiley & Sons, Inc., New York, New York, USA, pp. 1-326, 2000.

\* cited by examiner

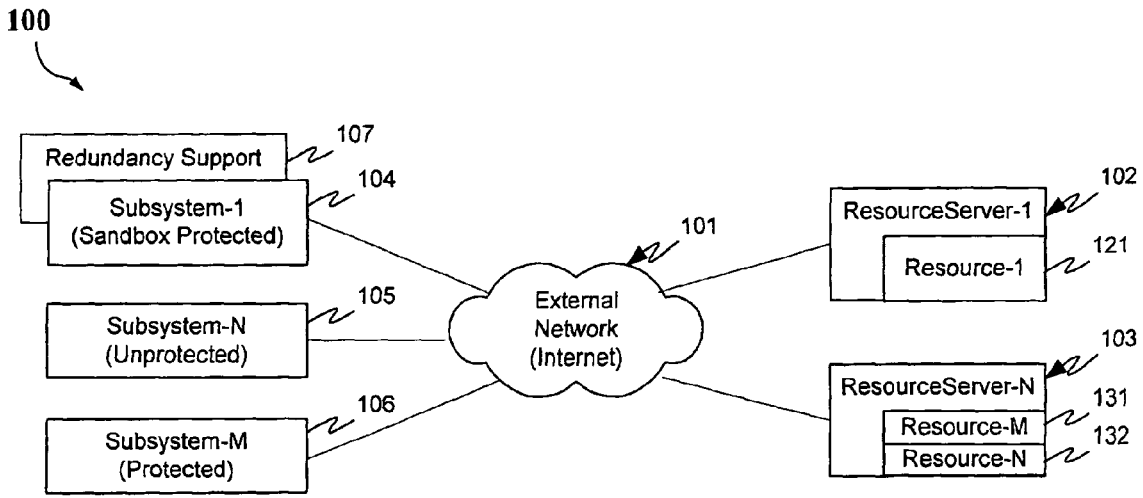


FIG. 1a

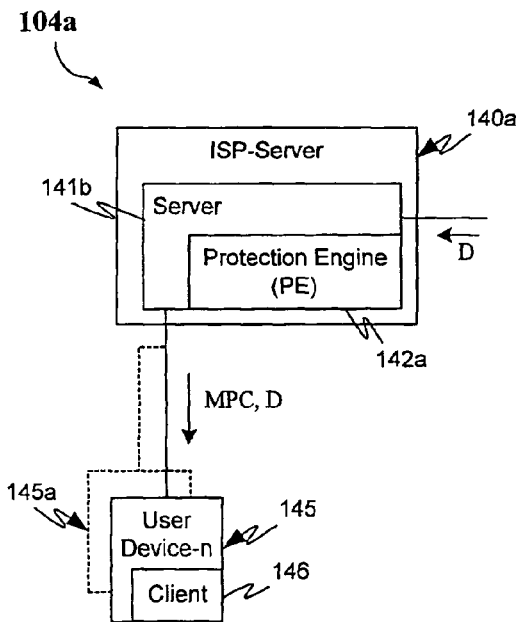


FIG. 1b

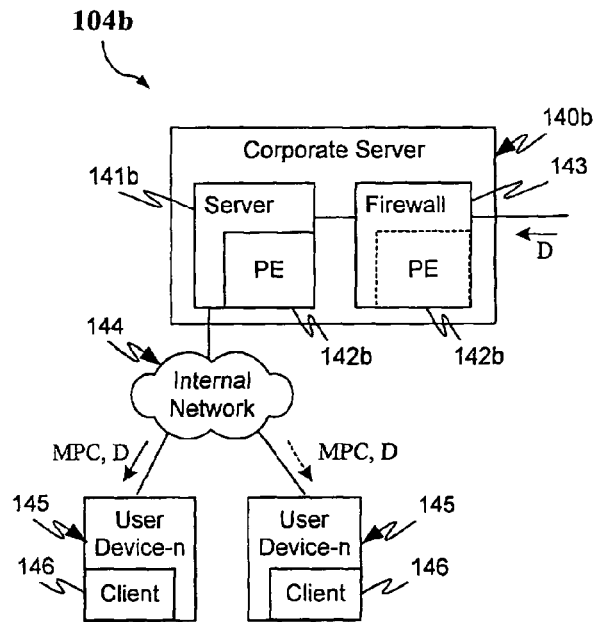


FIG. 1c

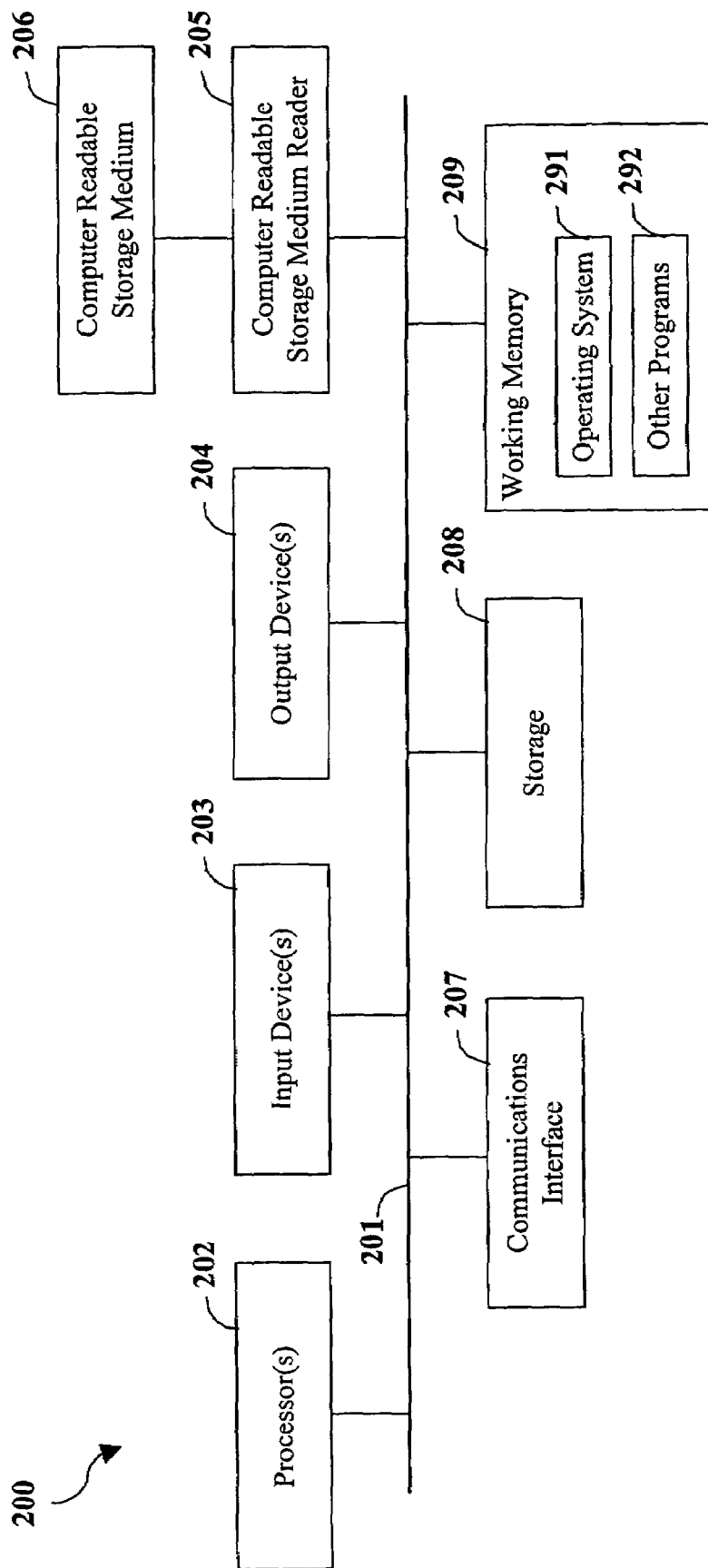
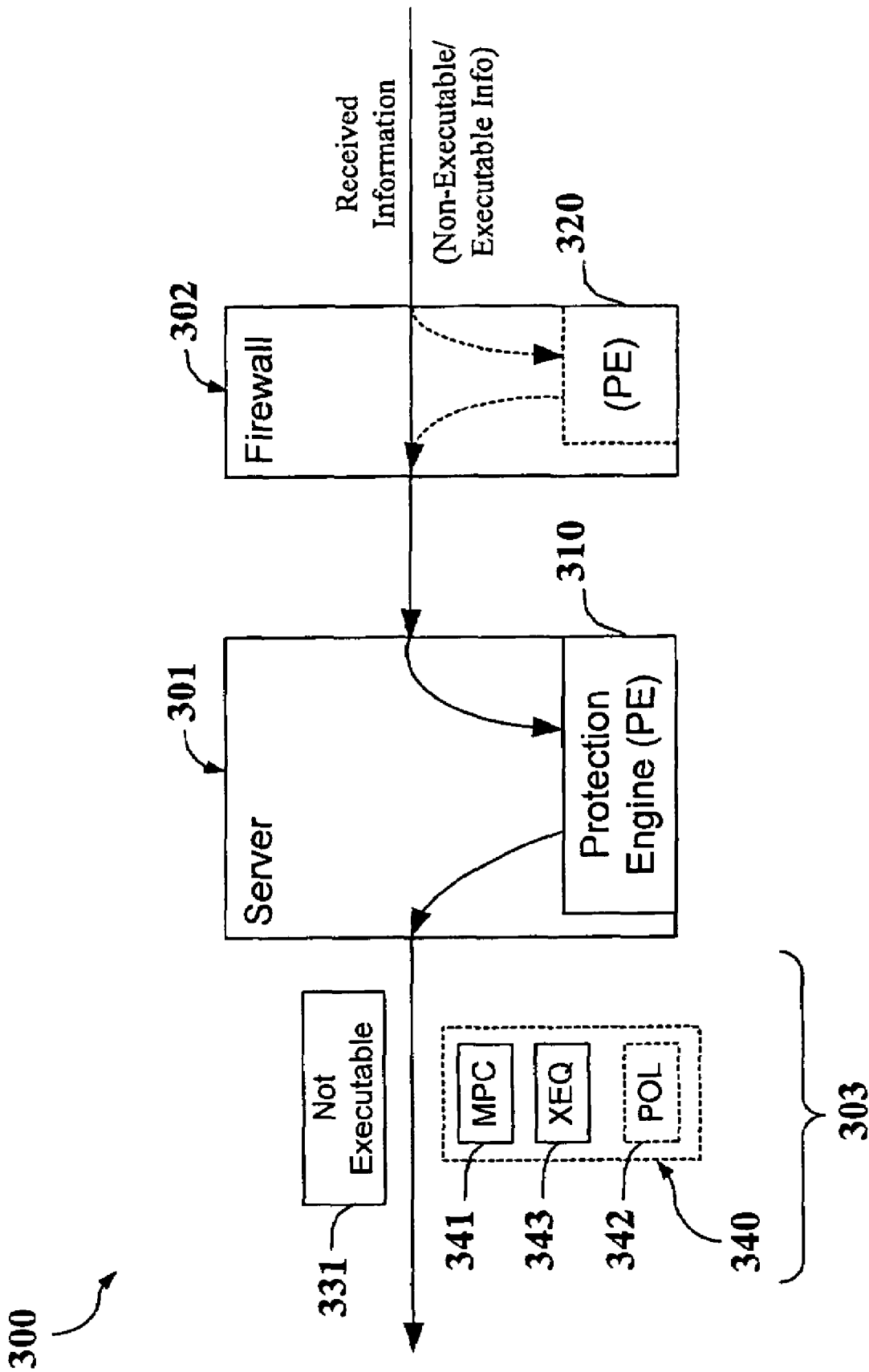


FIG. 2



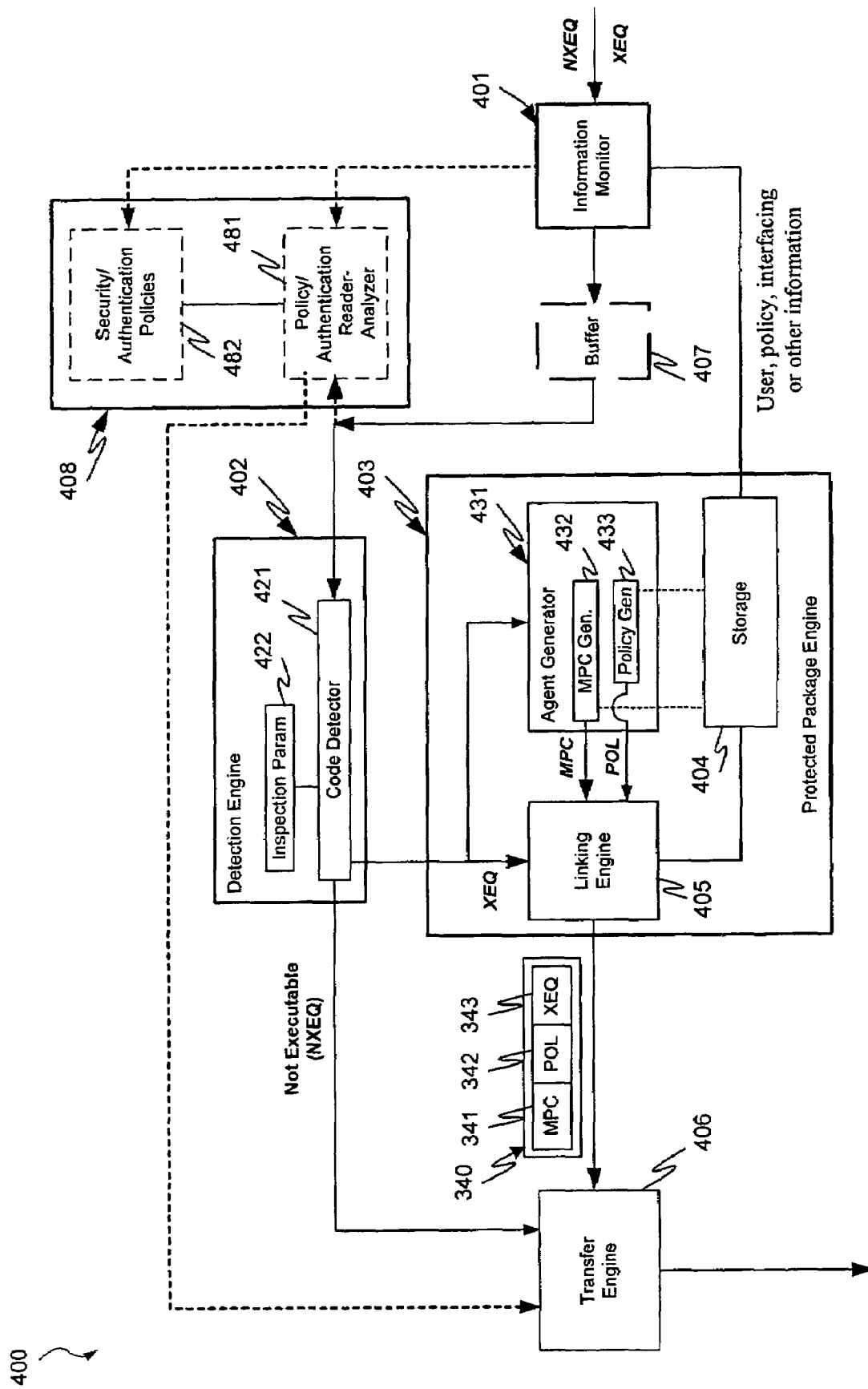


FIG. 4

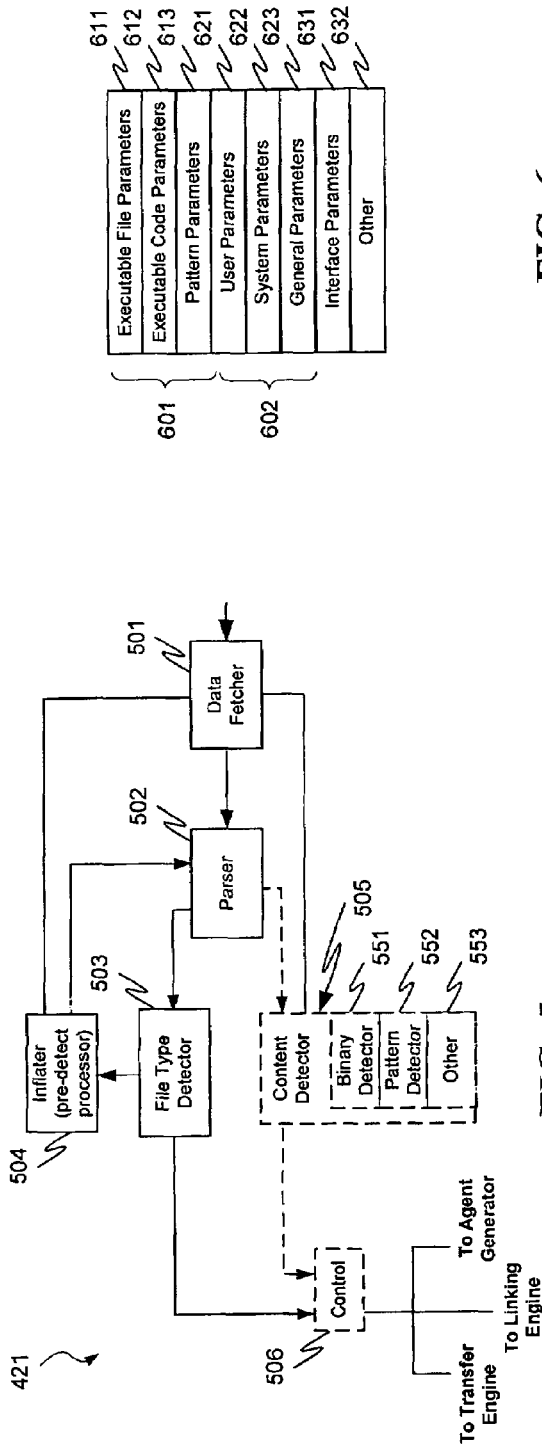


FIG. 6a

FIG. 5

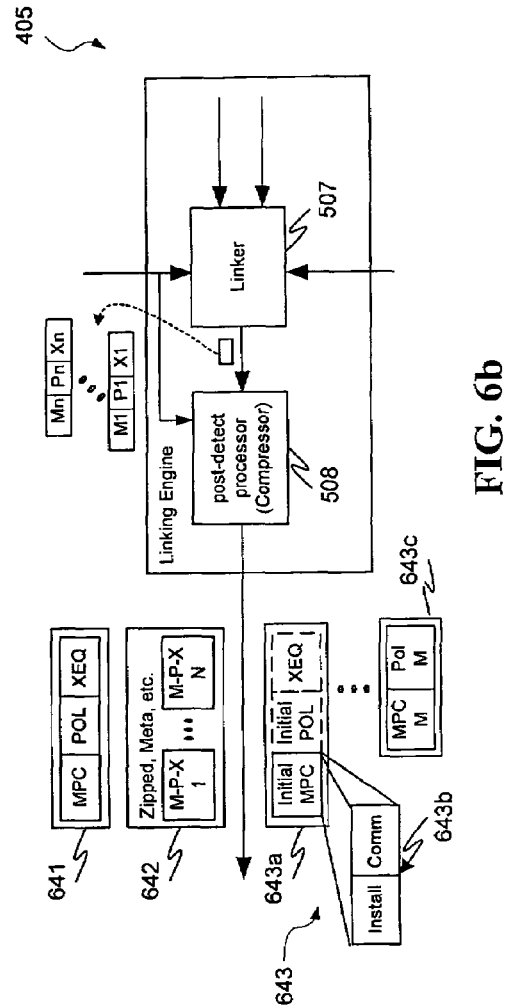


FIG. 6b



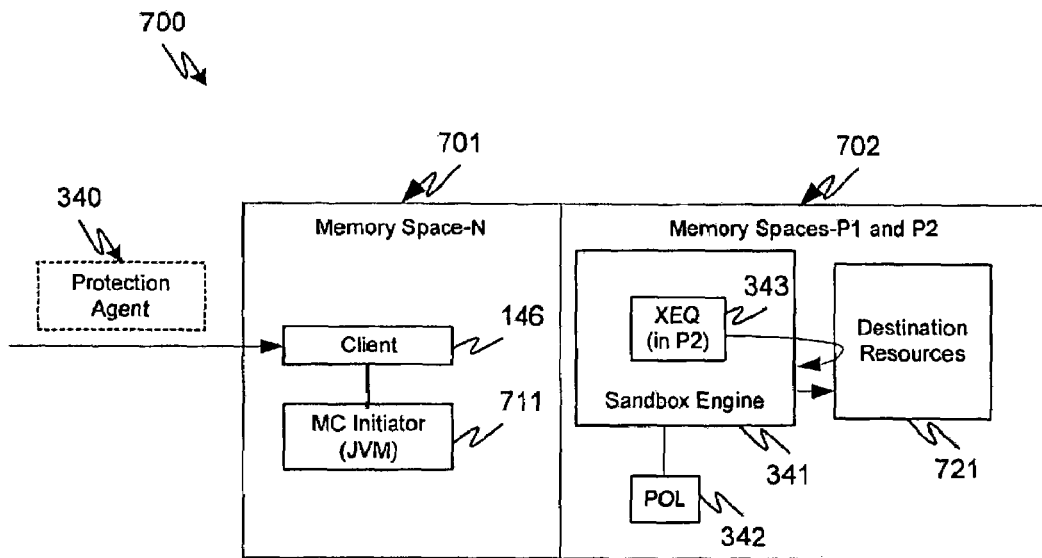


FIG. 7a

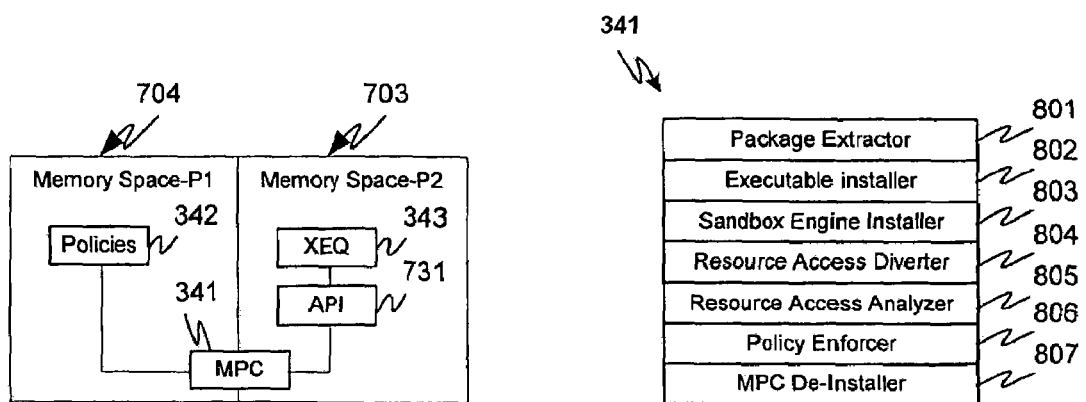


FIG. 7b

FIG. 8

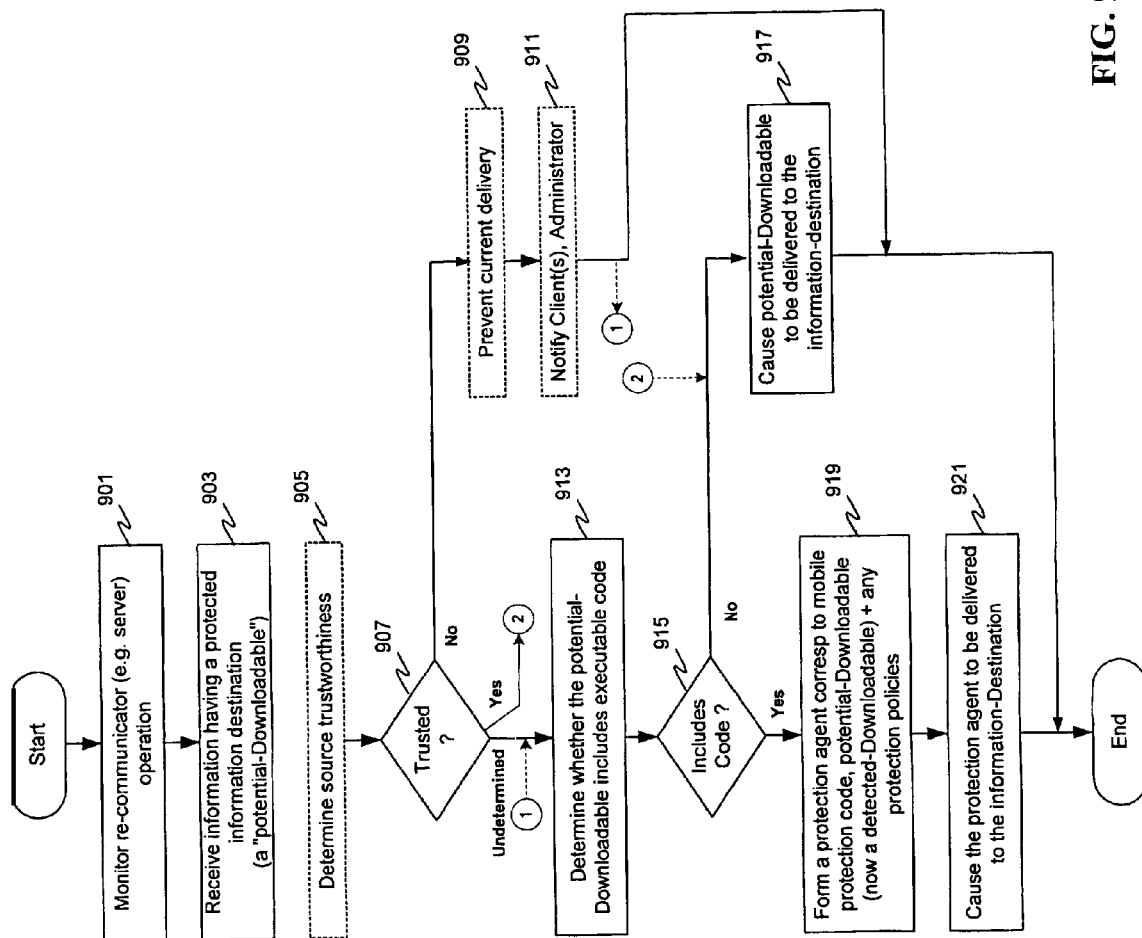


FIG. 9

913

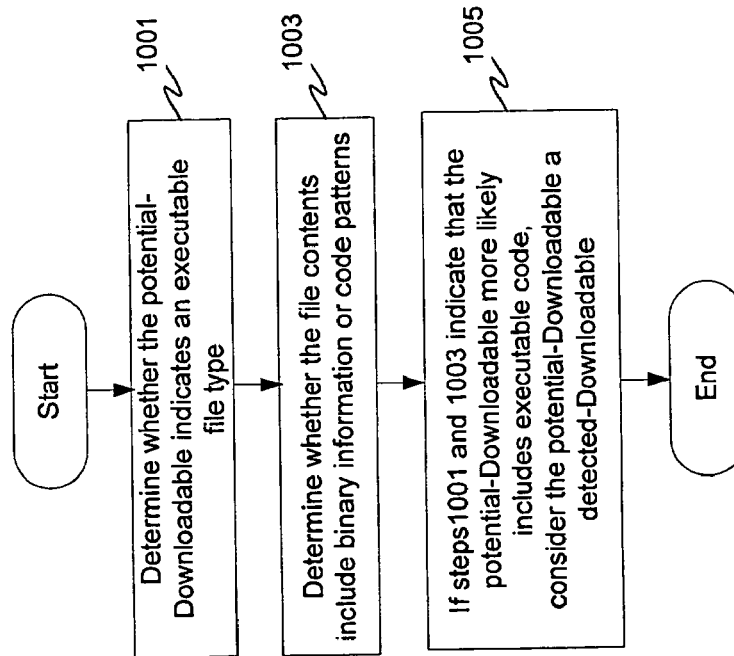


FIG. 10A

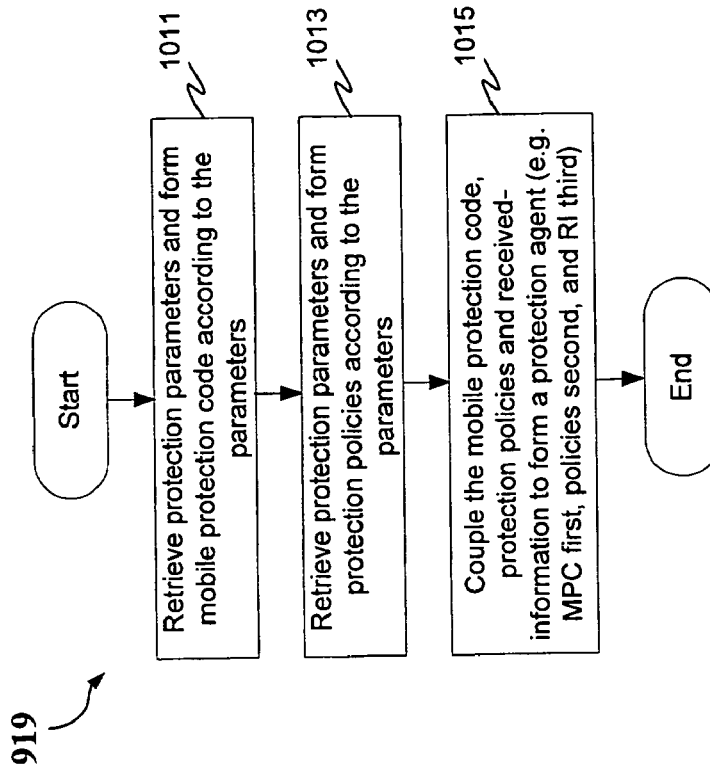


FIG. 10B

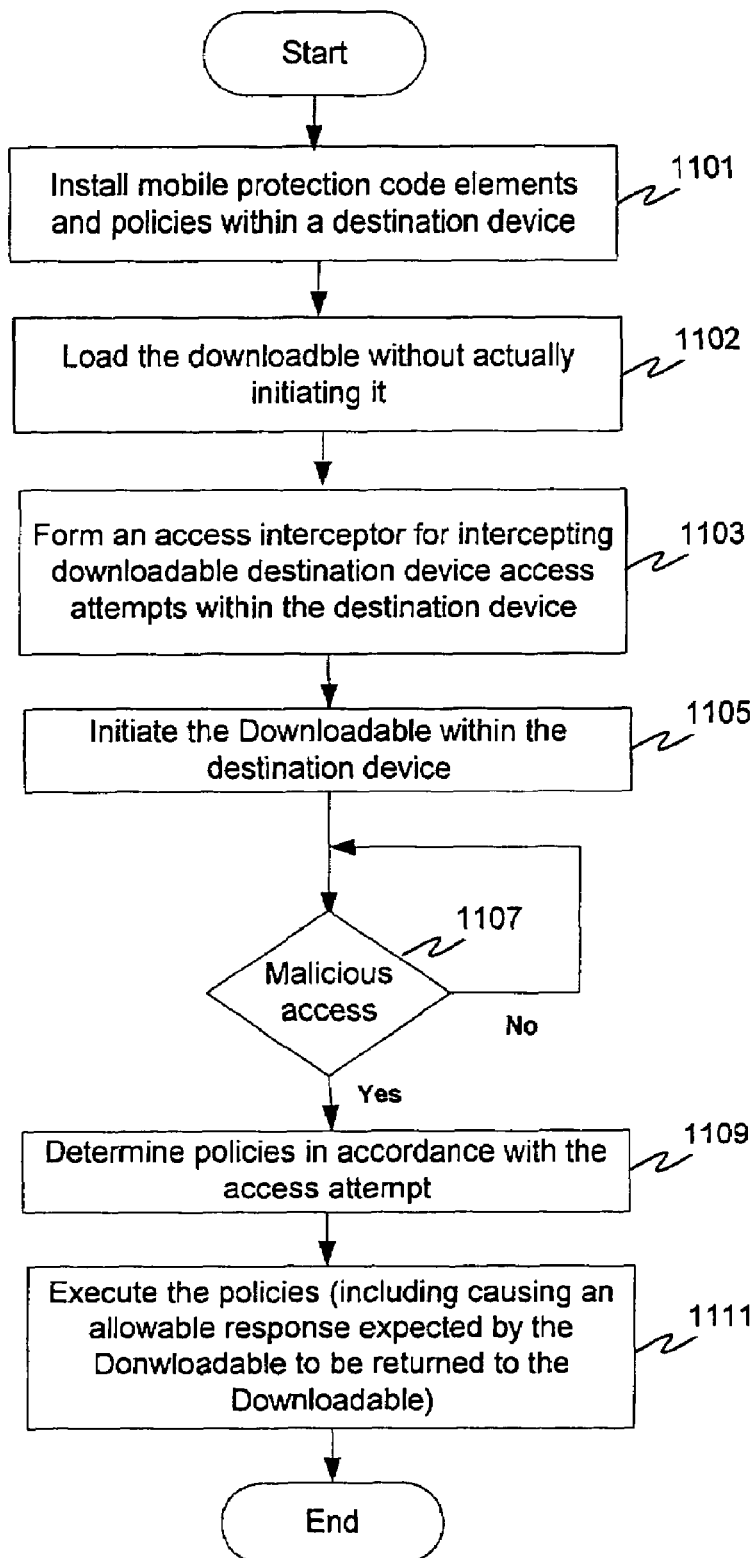
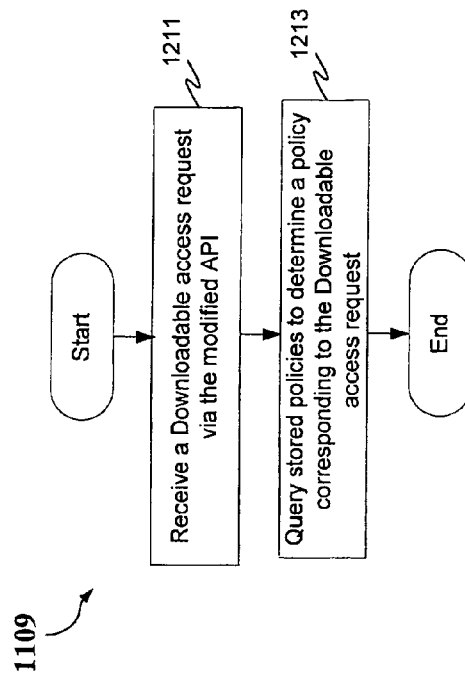
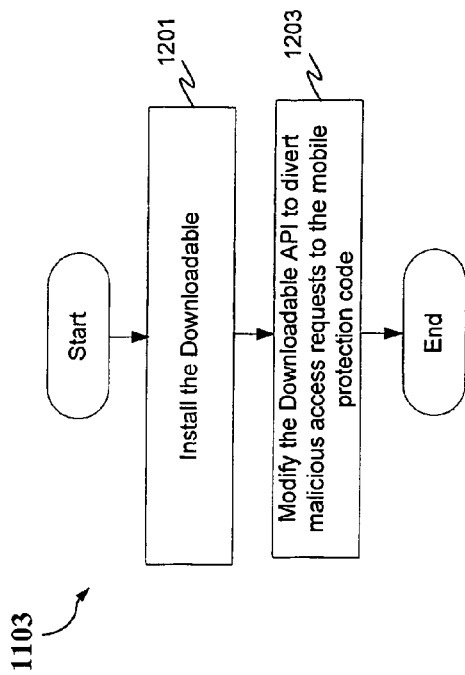


FIG. 11



US 7,613,926 B2

1

## METHOD AND SYSTEM FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

### PRIORITY REFERENCE TO RELATED APPLICATIONS

This application is a continuation of assignee's application Ser. No. 09/861,229, filed on May 17, 2001, now U.S. Pat. No. 7,058,822, entitled "Malicious Mobile Code Runtime Monitoring System And Methods", which is hereby incorporated by reference. U.S. application Ser. No. 09/861,229 claims benefit of provisional application Ser. No. 60/205,591, entitled "Computer Network Malicious Code Run-time Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al., which is hereby incorporated by reference. U.S. application Ser. No. 09/861,229 is also a Continuation-In-Part of U.S. patent application Ser. No. 09/539,667, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul, now U.S. Pat. No. 6,804,780, and hereby incorporated by reference, which is a continuation of assignee's patent application U.S. Ser. No. 08/964,388, filed on Nov. 6, 1997, now U.S. Pat. No. 6,092,194, also entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables" and hereby incorporated by reference. U.S. Ser. No. 09/861,229 is also a Continuation-In-Part of U.S. patent application Ser. No. 09/551,302, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul, now U.S. Pat. No. 6,480,962, which is hereby incorporated by reference.

### BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and methods for protecting network-connectable devices from undesirable downloadable operation.

#### 2. Description of the Background Art

Advances in networking technology continue to impact an increasing number and diversity of users. The Internet, for example, already provides to expert, intermediate and even novice users the informational, product and service resources of over 100,000 interconnected networks owned by governments, universities, nonprofit groups, companies, etc. Unfortunately, particularly the Internet and other public networks have also become a major source of potentially system-fatal or otherwise damaging computer code commonly referred to as "viruses."

Efforts to forestall viruses from attacking networked computers have thus far met with only limited success at best. Typically, a virus protection program designed to identify and remove or protect against the initiating of known viruses is installed on a network firewall or individually networked computer. The program is then inevitably surmounted by some new virus that often causes damage to one or more computers. The damage is then assessed and, if isolated, the new virus is analyzed. A corresponding new virus protection program (or update thereof) is then developed and installed to combat the new virus, and the new program operates successfully until yet another new virus appears—and so on. Of course, damage has already typically been incurred.

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable

2

information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only distributable components including "Java applets or ActiveX controls", and further does so using resource intensive and high bandwidth static Downloadable content and operational analysis, and modification of the Downloadable component; Shuang further fails to detect or protect against additional program code included within a tested Downloadable. U.S. Pat. No. 5,974,549 to Golan teaches a protection system that further focuses only on protecting against ActiveX controls and not other distributable components, let alone other Downloadable types. U.S. Pat. No. 6,167,520 to Touboul enables more accurate protection than Shuang or Golan, but lacks the greater flexibility and efficiency taught herein, as do Shuang and Golan.

Accordingly, there remains a need for efficient, accurate and flexible protection of computers and other network connectable devices from malicious Downloadables.

### SUMMARY OF THE INVENTION

The present invention provides protection systems and methods capable of Protecting a personal computer ("PC") or other persistently or even intermittently network accessible devices or processes from harmful, undesirable, suspicious or other "malicious" operations that might otherwise be effectuated by remotely operable code. While enabling the capabilities of prior systems, the present invention is not nearly so limited, resource intensive or inflexible, and yet enables more reliable protection. For example, remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software "components", such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others. Protection can also be provided in a distributed interactively, automatically or mixed configurable manner using protected client, server or other parameters, redirection, local/remote logging, etc., and other server/client based protection measures can also be separately and/or interoperably utilized, among other examples.

In one aspect, embodiments of the invention provide for determining, within one or more network "servers" (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a "Downloadable"). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

A protection engine according to an embodiment of the invention is operable within one or more network servers, firewalls or other network connectable information re-com-

US 7,613,926 B2

3

municating devices (as are referred to herein summarily one or more “servers” or “re-communicators”). The protection engine includes an information monitor for monitoring information received by the server, and a code detection engine for determining whether the received information includes executable code. The protection engine also includes a packaging engine for causing a sandboxed package, typically including mobile protection code and downloadable protection policies to be sent to a Downloadable-destination in conjunction with the received information, if the received information is determined to be a Downloadable.

A sandboxed package according to an embodiment of the invention is receivable by and operable with a remote Downloadable-destination. The sandboxed package includes mobile protection code (“MPC”) for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted. The sandboxed package also includes protection policies (operable alone or in conjunction with further Downloadable-destination stored or received policies/MPCs) for causing one or more predetermined operations to be performed if one or more undesirable operations of the Downloadable is/are intercepted. The sandboxed package can also include a corresponding Downloadable and can provide for initiating the Downloadable in a protective “sandbox”. The MPC/policies can further include a communicator for enabling further MPC/policy information or “modules” to be utilized and/or for event logging or other purposes.

A sandbox protection system according to an embodiment of the invention comprises an installer for enabling a received MPC to be executed within a Downloadable-destination (device/process) and further causing a Downloadable application program, distributable component or other received downloadable code to be received and installed within the Downloadable-destination. The protection system also includes a diverter for monitoring one or more operation attempts of the Downloadable, an operation analyzer for determining one or more responses to the attempts, and a security enforcer for effectuating responses to the monitored operations. The protection system can further include one or more security policies according to which one or more protection system elements are operable automatically (e.g. programmatically) or in conjunction with user intervention (e.g. as enabled by the security enforcer). The security policies can also be configurable/extensible in accordance with further downloadable and/or Downloadable-destination information.

A method according to an embodiment of the invention includes receiving downloadable information, determining whether the downloadable information includes executable code, and causing a mobile protection code and security policies to be communicated to a network client in conjunction with security policies and the downloadable information if the downloadable information is determined to include executable code. The determining can further provide multiple tests for detecting, alone or together, whether the downloadable information includes executable code.

A further method according to an embodiment of the invention includes forming a sandboxed package that includes mobile protection code (“MPC”), protection policies, and a received, detected-Downloadable, and causing the sandboxed package to be communicated to and installed by a receiving device or process (“user device”) for responding to one or more malicious operation attempts by the detected-Downloadable from within the user device. The MPC/policies can further include a base “module” and a “communica-

4

tor” for enabling further up/downloading of one or more further “modules” or other information (e.g. events, user/user device information, etc.).

Another method according to an embodiment of the invention includes installing, within a user device, received mobile protection code (“MPC”) and protection policies in conjunction with the user device receiving a downloadable application program, component or other Downloadable(s). The method also includes determining, by the MPC, a resource access attempt by the Downloadable, and initiating, by the MPC, one or more predetermined operations corresponding to the attempt. (Predetermined operations can, for example, comprise initiating user, administrator, client, network or protection system determinable operations, including but not limited to modifying the Downloadable operation, extricating the Downloadable, notifying a user/another, maintaining a local/remote log, causing one or more MPCs/policies to be downloaded, etc.)

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against without modifying the mobile code. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented. Embodiments further provide for thwarting efforts to bypass security measures (e.g. by “hiding” undesirable operation causing information within apparently inert or otherwise “friendly” downloadable information) and/or dividing or combining security measures for even greater flexibility and/or efficiency.

Embodiments also provide for determining protection policies that can be downloaded and/or ascertained from other security information (e.g. browser settings, administrative policies, user input, uploaded information, etc.). Different actions in response to different Downloadable operations, clients, users and/or other criteria are also enabled, and embodiments provide for implementing other security measures, such as verifying a downloadable source, certification, authentication, etc. Appropriate action can also be accomplished automatically (e.g. programmatically) and/or in conjunction with alerting one or more users/administrators, utilizing user input, etc. Embodiments further enable desirable Downloadable operations to remain substantially unaffected, among other aspects.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1a is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

FIG. 1b is a block diagram illustrating a network subsystem example in accordance with an embodiment of the invention;

FIG. 1c is a block diagram illustrating a further network subsystem example in accordance with an embodiment of the invention;

FIG. 2 is a block diagram illustrating a computer system in accordance with an embodiment of the invention;

FIG. 3 is a flow diagram broadly illustrating a protection system host according to an embodiment of the invention;

FIG. 4 is a block diagram illustrating a protection engine according to an embodiment of the invention;

## US 7,613,926 B2

5

FIG. 5 is a block diagram illustrating a content inspection engine according to an embodiment of the invention;

FIG. 6a is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. 6b is a flow diagram illustrating a linking engine use in conjunction with ordinary, compressed and distributable sandbox package utilization, according to an embodiment of the invention;

FIG. 7a is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. 7b is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. 7a, according to an embodiment of the invention;

FIG. 8 is a block diagram illustrating a mobile protection code according to an embodiment of the invention;

FIG. 9 is a flowchart illustrating a protection method according to an embodiment of the invention;

FIG. 10a is a flowchart illustrating method for determining if a potential-Downloadable includes or is likely to include executable code, according to an embodiment of the invention;

FIG. 10b is a flowchart illustrating a method for forming a protection agent, according to an embodiment of the invention;

FIG. 11 is a flowchart illustrating a method for protecting a Downloadable destination according to an embodiment of the invention;

FIG. 12a is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. 12b is a flowchart illustrating a method for implementing mobile protection policies according to an embodiment of the invention.

## DETAILED DESCRIPTION

In providing malicious mobile code runtime monitoring systems and methods, embodiments of the invention enable actually or potentially undesirable operations of even unknown malicious code to be efficiently and flexibly avoided. Embodiments provide, within one or more “servers” (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information and detecting whether the downloadable-information includes one or more instances of executable code (e.g. as with a Trojan horse, zip/meta file etc.). Embodiments also provide for separately or interoperably conducting additional security measures within the server, within a Downloadable-destination of a detected-Downloadable, or both. Embodiments further provide for causing mobile protection code (“MPC”) and downloadable protection policies to be communicated to, installed and executed within one or more received information destinations in conjunction with a detected-Downloadable. Embodiments also provide, within an information-destination, for detecting malicious operations of the detected-Downloadable and causing responses thereto in accordance with the protection policies (which can correspond to one or more user, Downloadable, source, destination, or other parameters), or further downloaded or downloadable-destination based policies (which can also be configurable or extensible). (Note that the term “or”, as used herein, is generally intended to mean “and/or” unless otherwise indicated.)

FIGS. 1a through 1c illustrate a computer network system 100 according to an embodiment of the invention. FIG. 1a broadly illustrates system 100, while FIGS. 1b and 1c illus-

6

trate exemplary protectable subsystem implementations corresponding with system 104 or 106 of FIG. 1a.

Beginning with FIG. 1a, computer network system 100 includes an external computer network 101, such as a Wide Area Network or “WAN” (e.g. the Internet), which is coupled to one or more network resource servers (summarily depicted as resource server-1 102 and resource server-N 103). Where external network 101 includes the Internet, resource servers 1-N (102, 103) might provide one or more resources including web pages, streaming media, transaction-facilitating information, program updates or other downloadable information, summarily depicted as resources 121, 131 and 132. Such information can also include more traditionally viewed “Downloadables” or “mobile code” (i.e. distributable components), as well as downloadable application programs or other further Downloadables, such as those that are discussed herein. (It will be appreciated that interconnected networks can also provide various other resources as well.)

Also coupled via external network 101 are subsystems 104-106. Subsystems 104-106 can, for example, include one or more servers, personal computers (“PCs”), smart appliances, personal information managers or other devices/processes that are at least temporarily or otherwise intermittently directly or indirectly connectable in a wired or wireless manner to external network 101 (e.g. using a dialup, DSL, cable modem, cellular connection, IR/RF, or various other suitable current or future connection alternatives). One or more of subsystems 104-106 might further operate as user devices that are connectable to external network 101 via an internet service provider (“ISP”) or local area network (“LAN”), such as a corporate intranet, or home, portable device or smart appliance network, among other examples.

FIG. 1a also broadly illustrates how embodiments of the invention are capable of selectively, modifiably or extensively providing protection to one or more determinable ones of networked subsystems 104-106 or elements thereof (not shown) against potentially harmful or other undesirable (“malicious”) effects in conjunction with receiving downloadable information. “Protected” subsystem 104, for example, utilizes a protection in accordance with the teachings herein, while “unprotected” subsystem-N 105 employs no protection, and protected subsystem-M 106 might employ one or more protections including those according to the teachings herein, other protection, or some combination.

System 100 implementations are also capable of providing protection to redundant elements 107 of one or more of subsystems 104-106 that might be utilized, such as backups, failsafe elements, redundant networks, etc. Where included, such redundant elements are also similarly protectable in a separate, combined or coordinated manner using embodiments of the present invention either alone or in conjunction with other protection mechanisms. In such cases, protection can be similarly provided singly, as a composite of component operations or in a backup fashion. Care should, however, be exercised to avoid potential repeated protection engine execution corresponding to a single Downloadable; such “chaining” can cause a Downloadable to operate incorrectly or not at all, unless a subsequent detection engine is configured to recognize a prior packaging of the Downloadable.

FIGS. 1b and 1c further illustrate, by way of example, how protection systems according to embodiments of the invention can be utilized in conjunction with a wide variety of different system implementations. In the illustrated examples, system elements are generally configurable in a manner commonly referred to as a “client-server” configuration, as is typically utilized for accessing Internet and many other network resources. For clarity sake, a simple client-



server configuration will be presumed unless otherwise indicated. It will be appreciated, however, that other configurations of interconnected elements might also be utilized (e.g. peer-peer, routers, proxy servers, networks, converters, gateways, services, network reconfiguring elements, etc.) in accordance with a particular application.

The FIG. 1*b* example shows how a suitable protected system 104*a* (which can correspond to subsystem-1 104 or subsystem-M 106 of FIG. 1) can include a protection-initiating host “server” or “re-communicator” (e.g. ISP server 140*a*), one or more user devices or “Downloadable-destinations” 145, and zero or more redundant elements (which elements are summarily depicted as redundant client device/process 145*a*). In this example, ISP server 140*a* includes one or more email, Internet or other servers 141*a*, or other devices or processes capable of transferring or otherwise “re-communicating” downloadable information to user devices 145. Server 141*a* further includes protection engine or “PE” 142*a*, which is capable of supplying mobile protection code (“MPC”) and protection policies for execution by client devices 145. One or more of user devices 145 can further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which MPC and protection policies are operable to protect user devices 145 from detrimental, undesirable or otherwise “malicious” operations of downloadable information also received by user device 145.

The FIG. 1*c* example shows how a further suitable protected system 104*b* can include, in addition to a “re-communicator”, such as server 142*b*, a firewall 143*c* (e.g. as is typically the case with a corporate intranet and many existing or proposed home/smart networks.) In such cases, a server 141*b* or firewall 143 can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server 141*b* and firewall 143, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

System 104*b* also includes internal network 144 and user devices 145. User devices 145 further include a respective one or more clients 146 for utilizing information received via server 140*a*, in accordance with which the MPCs or protection policies are operable. (As in the previous example, one or more of user devices 145 can also include or correspond with similarly protectable redundant system elements, which are not shown.)

It will be appreciated that the configurations of FIGS 1*a-1c* are merely exemplary. Alternative embodiments might, for example, utilize other suitable connections, devices or processes. One or more devices can also be configurable to operate as a network server, firewall, smart router, a resource server servicing deliverable third-party/manufacturer postings, a user device operating as a firewall/server, or other information-suppliers or intermediaries (i.e. as a “re-communicator” or “server”) for servicing one or more further interconnected devices or processes or interconnected levels of devices or processes. Thus, for example, a suitable protection engine host can include one or more devices or processes capable of providing or supporting the providing of mobile protection code or other protection consistent with the teachings herein. A suitable information-destination or “user device” can further include one or more devices or processes (such as email, browser or other clients) that are capable of receiving and initiating or otherwise hosting a mobile code execution.

FIG. 2 illustrates an exemplary computing system 200, that can comprise one or more of the elements of FIGS. 1*a* through 1*c*. While other application-specific alternatives might be utilized, it will be presumed for clarity sake that system 100 elements (FIGS. 1*a-c*) are implemented in hardware, software or some combination by one or more processing systems consistent therewith, unless otherwise indicated.

Computer system 200 comprises elements coupled via communication channels (e.g. bus 201) including one or more general or special purpose processors 202, such as a Pentium® or Power PC®, digital signal processor (“DSP”), etc. System 200 elements also include one or more input devices 203 (such as a mouse, keyboard, microphone, pen, etc.), and one or more output devices 204, such as a suitable display, speakers, actuators, etc., in accordance with a particular application.

System 200 also includes a computer readable storage media reader 205 coupled to a computer readable storage medium 206, such as a storage/memory device or hard or removable storage/memory media; such devices or media are further indicated separately as storage device 208 and memory 209, which can include hard disk variants, floppy/compact disk variants, digital versatile disk (“DVD”) variants, smart cards, read only memory, random access memory, cache memory, etc., in accordance with a particular application. One or more suitable communication devices 207 can also be included, such as a modem, DSL, infrared or other suitable transceiver, etc. for providing inter-device communication directly or via one or more suitable private or public networks that can include but are not limited to those already discussed.

Working memory further includes operating system (“OS”) elements and other programs, such as application programs, mobile code, data, etc. for implementing system 100 elements that might be stored or loaded therein during use. The particular OS can vary in accordance with a particular device, features or other aspects in accordance with a particular application (e.g. Windows, Mac, Linux, Unix or Palm OS variants, a proprietary OS, etc.). Various programming languages or other tools can also be utilized, such as C++, Java, Visual Basic, etc. As will be discussed, embodiments can also include a network client such as a browser or email client, e.g. as produced by Netscape, Microsoft or others, a mobile code executor such as an OS task manager, Java Virtual Machine (“JVM”), etc., and an application program interface (“API”), such as a Microsoft Windows or other suitable element in accordance with the teachings herein. (It will also become apparent that embodiments might also be implemented in conjunction with a resident application or combination of mobile code and resident application components.)

One or more system 200 elements can also be implemented in hardware, software or a suitable combination. When implemented in software (e.g. as an application program, object, downloadable, servlet, etc. in whole or part), a system 200 element can be communicated transitionally or more persistently from local or remote storage to memory (or cache memory, etc.) for execution, or another suitable mechanism can be utilized, and elements can be implemented in compiled or interpretive form. Input, intermediate or resulting data or functional elements can further reside more transitionally or more persistently in a storage media, cache or more persistent volatile or non-volatile memory, (e.g. storage device 207 or memory 208) in accordance with a particular application.

FIG. 3 illustrates an interconnected re-communicator 300 generally consistent with system 140*b* of FIG. 1, according to an embodiment of the invention. As with system 140*b*, system

**300** includes a server **301**, and can also include a firewall **302**. In this implementation, however, either server **301** or firewall **302** (if a firewall is used) can further include a protection engine (**310** or **320** respectively). Thus, for example, an included firewall can process received information in a conventional manner, the results of which can be further processed by protection engine **310** of server **301**, or information processed by protection engine **320** of an included firewall **302** can be processed in a conventional manner by server **301**. (For clarity sake, a server including a singular protection engine will be presumed, with or without a firewall, for the remainder of the discussion unless otherwise indicated. Note, however, that other embodiments consistent with the teachings herein might also be utilized.)

FIG. 3 also shows how information received by server **301** (or firewall **302**) can include non-executable information, executable information or a combination of non-executable and one or more executable code portions (e.g. so-called Trojan horses that include a hostile Downloadable within a friendly one, combined, compressed or otherwise encoded files, etc.). Particularly such combinations will likely remain undetected by a firewall or other more conventional protection systems. Thus, for convenience, received information will also be referred to as a “potential-Downloadable”, and received information found to include executable code will be referred to as a “Downloadable” or equivalently as a “detected-Downloadable” (regardless of whether the executable code includes one or more application programs, distributable “components” such as Java, ActiveX, add-in, etc.).

Protection engine **310** provides for detecting whether received potential-Downloadables include executable code, and upon such detection, for causing mobile protection code (“MPC”) to be transferred to a device that is a destination of the potential-Downloadable (or “Downloadable-destination”). Protection engine **310** can also provide protection policies in conjunction with the MPC (or thereafter as well), which MPC/policies can be automatically (e.g. programmatically) or interactively configurable in accordance user, administrator, downloadable source, destination, operation, type or various other parameters alone or in combination (see below). Protection engine **310** can also provide or operate separately or interoperably in conjunction with one or more of certification, authentication, downloadable tagging, source checking, verification, logging, diverting or other protection services via the MPC, policies, other local/remote server or destination processing, etc. (e.g. which can also include protection mechanisms taught by the above-noted prior applications; see FIG. 4).

Operationally, protection engine **310** of server **301** monitors information received by server **301** and determines whether the received information is deliverable to a protected destination, e.g. using a suitable monitor/data transfer mechanism and comparing a destination-address of the received information to a protected destination set, such as a protected destinations list, array, database, etc. (All deliverable information or one or more subsets thereof might also be monitored.) Protection engine **310** further analyzes the potential-Downloadable and determines whether the potential-Downloadable includes executable code. If not, protection engine **310** enables the not executable potential-Downloadable **331** to be delivered to its destination in an unaffected manner.

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine **310** also causes mobile protection code or “MPC” **341** to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the

detected-Downloadable **343** (see below). Protection engine **310** further causes downloadable protection policies **342** to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies **342** provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

(One or more “checked”, tag, source, destination, type, detection or other security result indicators, which are not shown, can also be provided as corresponding to determined non-Downloadables or Downloadables, e.g. for testing, logging, further processing, further identification tagging or other purposes in accordance with a particular application.)

Further MPCs, protection policies or other information are also deliverable to a the same or another destination, for example, in accordance with communication by an MPC/protection policies already delivered to a downloadable-destination. Initial or subsequent MPCs/policies can further be selected or configured in accordance with a Downloadable-destination indicated by the detected-Downloadable, destination-user or administrative information, or other information providable to protection engine **310** by a user, administrator, user system, user system examination by a communicated MPC, etc. (Thus, for example, an initial MPC/policies can also be initially provided that are operable with or optimized for more efficient operation with different Downloadable-destinations or destination capabilities.)

While integrated protection constraints within the MPC might also be utilized, providing separate protection policies has been found to be more efficient, for example, by enabling more specific protection constraints to be more easily updated in conjunction with detected-Downloadable specifics, post-download improvements, testing, etc. Separate policies can further be more efficiently provided (e.g. selected, modified, instantiated, etc.) with or separately from an MPC, or in accordance with the requirements of a particular user, device, system, administration, later improvement, etc., as might also be provided to protection engine **310** (e.g. via user/MPC uploading, querying, parsing a Downloadable, or other suitable mechanism implemented by one or more servers or Downloadable-destinations).

(It will also become apparent that performing executable code detection and communicating to a downloadable-Destination an MPC and any applicable policies as separate from a detected-Downloadable is more accurate and far less resource intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.)

System **300** enables a single or extensible base-MPC to be provided, in anticipation or upon receipt of a first Downloadable, that is utilized thereafter to provide protection of one or more Downloadable-destinations. It is found, however, that providing an MPC upon each detection of a Downloadable (which is also enabled) can provide a desirable combination of configurability of the MPC/policies and lessened need for management (e.g. given potentially changing user/destination needs, enabling testing, etc.).

Providing an MPC upon each detection of a Downloadable also facilitates a lessened demand on destination resources, e.g. since information-destination resources used in executing the MPC/policies can be re-allocated following such use. Such alternatives can also be selectively, modifiably or extensibly provided (or further in accordance with other application-specific factors that might also apply.) Thus, for example, a base-MPC or base-policies might be provided to a

US 7,613,926 B2

11

user device that is/are extensible via additionally downloadable “modules” upon server 301 detection of a Downloadable deliverable to the same user device, among other alternatives.

In accordance with a further aspect of the invention, it is found that improved efficiency can also be achieved by causing the MPC to be executed within a Downloadable-destination in conjunction with, and further, prior to initiation of the detected Downloadable. One mechanism that provides for greater compatibility and efficiency in conjunction with conventional client-based Downloadable execution is for a protection engine to form a sandboxed package 340 including MPC 341, the detected-Downloadable 343 and any policies 342. For example, where the Downloadable is a binary executable to be executed by an operating system, protection engine 310 forms a protected package by concatenating, within sandboxed package 340, MPC 341 for delivery to a Downloadable-destination first, followed by protection policies 342 and Downloadable 343. (Concatenation or techniques consistent therewith can also be utilized for providing a protecting package corresponding to a Java applet for execution by a JVM of a Downloadable-destination, or with regard to ActiveX controls, add-ins or other distributable components, etc.)

The above concatenation or other suitable processing will result in the following. Upon receipt of sandboxed package 340 by a compatible browser, email or other destination-client and activating of the package by a user or the destination-client, the operating system (or a suitable responsively initiated distributed component host) will attempt to initiate sandboxed package 340 as a single Downloadable. Such processing will, however, result in initiating the MPC 341 and—in accordance with further aspects of the invention the MPC will initiate the Downloadable in a protected manner, further in accordance with any applicable included or further downloaded protection policies 342. (While system 300 is also capable of ascertaining protection policies stored at a Downloadable-destination, e.g. by poll, query, etc. of available destination information, including at least initial policies within a suitable protecting package is found to avoid associated security concerns or inefficiencies.)

Turning to FIG. 4, a protection engine 400 generally consistent with protection engine 310 (or 320) of FIG. 3 is illustrated in accordance with an embodiment of the invention. Protection engine 400 comprises information monitor 401, detection engine 402, and protected packaging engine 403, which further includes agent generator 431, storage 404, linking engine 405, and transfer engine 406. Protection engine 400 can also include a buffer 407, for temporarily storing a received potential-Downloadable, or one or more systems for conducting additional authentication, certification, verification or other security processing (e.g. summarily depicted as security system 408) Protection engine 400 can further provide for selectively re-directing, further directing, logging, etc. of a potential/detected Downloadable or information corresponding thereto in conjunction with detection, other security, etc., in accordance with a particular application.

(Note that FIG. 4, as with other figures included herein, also depicts exemplary signal flow arrows; such arrows are provided to facilitate discussion, and should not be construed as exclusive or otherwise limiting.)

Information monitor 401 monitors potential-Downloadables received by a host server and provides the information via buffer 407 to detection engine 402 or to other system 400 elements. Information monitor 401 can be configured to monitor host server download operations in conjunction with a user or a user-device that has logged-on to the server, or to

12

receive information via a server operation hook, servlet, communication channel or other suitable mechanism.

Information monitor 401 can also provide for transferring, to storage 404 or other protection engine elements, configuration information including, for example, user, MPC, protection policy, interfacing or other configuration information (e.g. see FIG. 6). Such configuration information monitoring can be conducted in accordance with a user/device logging onto or otherwise accessing a host server, via one or more of configuration operations, using an applet to acquire such information from or for a particular user, device or devices, via MPC/policy polling of a user device, or via other suitable mechanisms.

Detection engine 402 includes code detector 421, which receives a potential-Downloadable and determines, more suitably in conjunction with inspection parameters 422, whether the potential-Downloadable includes executable code and is thus a “detected-Downloadable”. (Code detector 421 can also include detection processors for performing file decompression or other “decoding”, or such detection-facilitating processing as decryption, utilization/support of security system 408, etc. in accordance with a particular application.)

Detection engine 402 further transfers a detected-downloadable (“XEQ”) to protected packaging engine 403 along with indicators of such detection, or a determined non-executable (“NXEQ”) to transfer engine 406. (Inspection parameters 422 enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. 5). Detection engine 402 can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can farther remain resident at a user device with further modules downloaded upon or even after delivery of a sandboxed package. Such distribution can also be provided in a configurable manner, such that delivery of a complete package or partial packages are automatically or interactively determinable in accordance with user/administrative preferences/policies, among other examples.

Packaging engine 403 provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable. In this example, packaging engine 403 includes agent generator 431, storage 404 and linking engine 405.

Agent generator 431 includes an MPC generator 432 and a protection policy generator 433 for “generating” an MPC and a protection policy (or set of policies) respectively upon receiving one or more “generate MPC/policy” indicators from detection engine 402, indicating that a potential-Downloadable is a detected-Downloadable. MPC generator 432 and protection policy generator 433 provide for generating MPCs and protection policies respectively in accordance with parameters retrieved from storage 404. Agent generator 431 is further capable of providing multiple MPCs/policies, for example, the same or different MPCs/policies in accordance with protecting ones of multiple executables within a zip file, or for providing initial MPCs/policies and then further MPCs/policies or MPC/policy “modules” as initiated by further indicators such as given above, via an indicator of an already downloaded MPC/policy or via other suitable mechanisms.

US 7,613,926 B2

13

(It will be appreciated that pre-constructed MPCs/policies or other processing can also be utilized, e.g. via retrieval from storage **404**, but with a potential decrease in flexibility.)

MPC generator **432** and protection policy generator **433** are further configurable. Thus, for example, more generic MPCs/policies can be provided to all or a grouping of serviced destination-devices (e.g. in accordance with a similarly configured/administered intranet), or different MPCs/policies that can be configured in accordance with one or more of user, network administration, Downloadable-destination or other parameters (e.g. see FIG. 6). As will become apparent, a resulting MPC provides an operational interface to a destination device/process. Thus, a high degree of flexibility and efficiency is enabled in providing such an operational interface within different or differently configurable user devices/processes or other constraints.

Such configurability further enables particular policies to be utilized in accordance with a particular application (e.g. particular system uses, access limitations, user interaction, treating application programs or Java components from a particular known source one way and unknown source ActiveX components, or other considerations). Agent generator **431** further transfers a resulting MPC and protection policy pair to linking engine **405**.

Linking engine **405** provides for forming from received component elements (see above) a sandboxed package that can include one or more initial or complete MPCs and applicable protection policies, and a Downloadable, such that the sandboxed package will protect a receiving Downloadable-destination from malicious operation by the Downloadable. Linking engine **405** is implementable in a static or configurable manner in accordance, for example, with characteristics of a particular user device/process stored intermittently or more persistently in storage **404**. Linking engine **405** can also provide for restoring a Downloadable, such as a compressed, encrypted or otherwise encoded file that has been decompressed, decrypted or otherwise decoded via detection processing (e.g. see FIG. 6b).

It is discovered, for example, that the manner in which the Windows OS initiates a binary executable or an ActiveX control can be utilized to enable protected initiation of a detected-Downloadable. Linking engine **405** is, for example, configurable to form, for an ordinary single-executable Downloadable (e.g. an application program, applet, etc.) a sandboxed package **340** as a concatenation of ordered elements including an MPC **341**, applicable policies **342** and the Downloadable or "XEQ" **343** (e.g. see FIG. 4).

Linking engine **405** is also configurable to form, for a Downloadable received by a server as a compressed single or multiple-executable Downloadable such as a zipped or meta file, a protecting package **340** including one or more MPCs, applicable policies and the one or more included executables of the Downloadable. For example, a sandboxed package can be formed in which a single MPC and policies precede and thus will affect all such executables as a result of inflating and installation. An MPC and applicable policies can also, for example, precede each executable, such that each executable will be separately sandboxed in the same or a different manner according to MPC/policy configuration (see above) upon inflation and installation. (See also FIGS. 5 and 6) Linking engine is also configurable to form an initial MPC, MPC-policy or sandboxed package (e.g. prior to upon receipt of a downloadable) or an additional MPC, MPC-policy or sandboxed package (e.g. upon or following receipt of a downloadable), such that suitable MPCs/policies can be provided to a Downloadable-destination or other destination in a more distributed manner. In this way, requisite bandwidth or destina-

14

tion resources can be minimized (via two or more smaller packages) in compromise with latency or other considerations raised by the additional required communication.

A configurable linking engine can also be utilized in accordance with other requirements of particular devices/processes, further or different elements or other permutations in accordance with the teachings herein. (It might, for example be desirable to modify the ordering of elements, to provide one or more elements separately, to provide additional information, such as a header, etc., or perform other processing in accordance with a particular device, protocol or other application considerations.)

Policy/authentication reader-analyzer **481** summarily depicts other protection mechanisms that might be utilized in conjunction with Downloadable detection, such as already discussed, and that can farther be configurable to operate in accordance with policies or parameters (summarily depicted by security/authentication policies **482**). Integration of such further protection in the depicted configuration, for example, enables a potential-Downloadable from a known unfriendly source, a source failing authentication or a provided-source that is confirmed to be fictitious to be summarily discarded, otherwise blocked, flagged, etc. (with or without further processing). Conversely, a potential-Downloadable from a known friendly source (or one confirmed as such) can be transferred with or without further processing in accordance with particular application considerations. (Other configurations including pre or post Downloadable detection mechanisms might also be utilized.)

Finally, transfer engine **406** of protection agent engine **303** provides for receiving and causing linking engine **405** (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine **406** is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine **406** can also be provided in a more configurable manner, such as was already discussed for other system **400** elements. (Any one or more of system **400** elements might be configurably implemented in accordance with a particular application.) Transfer engine **406** can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. 5 with reference to FIG. 4, a code detector **421** example is illustrated in accordance with an embodiment of the invention. As shown, code detector **421** includes data fetcher **501**, parser **502**, file-type detector **503**, inflater **504** and control **506**; other depicted elements. While implementable and potentially useful in certain instances, are found to require substantial overhead, to be less accurate in certain instances (see above) and are not utilized in a present implementation; these will be discussed separately below. Code detector elements are further configurable in accordance with stored parameters retrievable by data fetcher **501**. (A coupling between data fetcher **501** and control **506** has been removed for clarity sake.)

Data fetcher **501** provides for retrieving a potential-Downloadable or portions thereof stored in buffer **407** or parameters from storage **404**, and communicates such information or parameters to parser **502**. Parser **502** receives a potential-Downloadable or portions thereof from data fetcher **501** and isolates potential-Downloadable elements, such as file headers, source, destination, certificates, etc. for use by further processing elements.

File type detector **502** receives and determines whether the potential-Downloadable (likely) is or includes an executable file type. File-reader **502** can, for example, be configured to analyze a received potential-Downloadable for a file header,

which is typically included in accordance with conventional data transfer protocols, such as a portable executable or standard “.exe” file format for Windows OS application programs, a Java class header for Java applets, and so on for other applications, distributed components, etc. “Zipped”, meta or other compressed files, which might include one or more executables, also typically provide standard single or multi-level headers that can be read and used to identify included executable code (or other included information types). File type detector **502** is also configurable for analyzing potential-Downloadables for all potential file type delimiters or a more limited subset of potential file type delimiters (e.g. “.exe” or “.com” in conjunction with a DOS or Microsoft Windows OS Downloadable-destination).

Known file type delimiters can, for example, be stored in a more temporary or more persistent storage (e.g. storage **404** of FIG. **4**) which file type detector **502** can compare to a received potential-Downloadable. (Such delimiters can thus also be updated in storage **404** as a new file type delimiter is provided, or a more limited subset of delimiters can also be utilized in accordance with a particular Downloadable-destination or other considerations of a particular application.) File type detector **502** further transfers to controller **506** a detected file type indicator indicating that the potential-Downloadable includes or does not include (i.e. or likely include) an executable file type.

In this example, the aforementioned detection processor is also included as pre-detection processor or, more particularly, a configurable file inflater **504**. File inflater **504** provides for opening or “inflating” compressed files in accordance with a compressed file type received from file type detector **503** and corresponding file opening parameters received from data fetcher **501**. Where a compressed file (e.g. a meta file) includes nested file type information not otherwise reliably provided in an overall file header or other information, inflater **504** returns such information to parser **502**. File inflater **504** also provides any now-accessible included executables to control **506** where one or more included files are to be separately packaged with an MPC or policies.

Control **506**, in this example, operates in accordance with stored parameters and provides for routing detected non-Downloadables or Downloadables and control information, and for conducting the aforementioned distributed downloading of packages to Downloadable-destinations. In the case of a non-Downloadable, for example, control **506** sends the non-Downloadable to transfer engine **406** (FIG. **4**) along with any indicators that might apply. For an ordinary single-executable Downloadable, control **506** sends control information to agent generator **431** and the Downloadable to linking engine **405** along with any other applicable indicators (see **641** of FIG. **6b**). Control **506** similarly handles a compressed single-executable Downloadable or a multiple downloadable to be protected using a single sandboxed package. For a multiple-executable Downloadable, control **506** sends control information for each corresponding executable to agent generator **431**, and sends the executable to linking engine **405** along with controls and any applicable indicators, as in **643b** of FIG. **6b**. (The above assumes, however, that distributed downloading is not utilized; when used according to applicable parameters control **506** also operates in accordance with the following.)

Control **506** conducts distributed protection (e.g. distributed packaging) by providing control signals to agent generator **431**, linking engine **405** and transfer engine **406**. In the present example, control **506** initially sends controls to agent generator **431** and linking engine **405** (FIG. **4**) causing agent generator to generate an initial MPC and initial policies, and

sends control and a detected-Downloadable to linking engine **405**. Linking engine **405** forms an initial sandboxed package, which transfer engine causes (in conjunction with further controls) to be downloaded to the Downloadable destination (**643a** of FIG. **6b**). An initial MPC within the sandboxed package includes an installer and a communicator and performs installation as indicated below. The initial MPC also communicates via the communicator controls to control **506** (FIG. **5**) in response to which control **506** similarly causes generation of MPC-M and policy-M modules **643c**, which linking engine **405** links and transfer engine **406** causes to be sent to the Downloadable destination, and so on for any further such modules.

(It will be appreciated, however, that an initial package might be otherwise configured or sent prior to receipt of a Downloadable in accordance with configuration parameters or user interaction. Information can also be sent to other user devices, such as that of an administrator. Further MPCs/policies might also be coordinated by control **506** or other elements, or other suitable mechanisms might be utilized in accordance with the teachings herein.)

Regarding the remaining detection engine elements illustrated in FIG. **5**, where content analysis is utilized, parser **502** can also provide a Downloadable or portions thereof to content detector **505**. Content detector **505** can then provide one or more content analyses. Binary detector **551**, for example, performs detection of binary information; pattern detector **552** further analyzes the Downloadable for patterns indicating executable code, or other detectors can also be utilized. Analysis results therefrom can be used in an absolute manner, where a first testing result indicating executable code confirms Downloadable detection, which result is then sent to control **506**. Alternatively, however, composite results from such analyses can also be sent to control **506** for evaluation. Control **506** can further conduct such evaluation in a summary manner (determining whether a Downloadable is detected according to a majority or minimum number of indicators), or based on a weighting of different analysis results. Operation then continues as indicated above. (Such analysis can also be conducted in accordance with aspects of a destination user device or other parameters.) FIG. **6a** illustrates more specific examples of indicators/parameters and known (or “knowledge base”) elements that can be utilized to facilitate the above-discussed system **400** configurability and detection. For clarity sake, indicators, parameters and knowledge base elements are combined as indicated “parameters.” It will be appreciated, however, that the particular parameters utilized can differ in accordance with a particular application, and indicators, parameters or known elements, where utilized, can vary and need not correspond exactly with one another. Any suitable explicit or referencing list, database or other storage structure(s) or storage structure configuration(s) can also be utilized to implement a suitable user/device based protection scheme, such as in the above examples, or other desired protection schema.

Executable parameters **601** comprise, in accordance with the above examples, executable file type parameters **611**, executable code parameters **612** and code pattern parameters **613** (including known executable file type indicators, header/code indicators and patterns respectively, where code patterns are utilized). Use parameters **602** further comprise user parameters **621**, system parameters **622** and general parameters **623** corresponding to one or more users, user classifications, user-system correspondences or destination system, device or processes, etc. (e.g. for generating corresponding MPCs/policies, providing other protection, etc.). The remaining parameters include interface parameters **631** for provid-

ing MPC/policy (or further) configurability in accordance with a particular device or for enabling communication with a device user (see below), and other parameters **632**.

FIG. *6b* illustrates a linking engine **405** according to an embodiment of the invention. As already discussed, linking engine **405** includes a linker for combining MPCs, policies or agents via concatenation or other suitable processing in accordance with an OS, JVM or other host executor or other applicable factors that might apply. Linking engine **405** also includes the aforementioned post-detection processor which, in this example, comprises a compressor **508**. As noted, compressor **508** receives linked elements from linker **507** and, where a potential-Downloadable corresponds to a compressed file that was inflated during detection, re-forms the compressed file. (Known file information can be provided via configuration parameters, substantially reversal of inflating or another suitable method.) Encryption or other post-detection processing can also be conducted by linking engine **508**.

FIGS. *7a*, *7b* and *8* illustrate a “sandbox protection” system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. *7a*, a client **146** receiving sandbox package **340** will “recognize” sandbox package **340** as a (mobile) executable and cause a mobile code installer **711** (e.g. an OS loader, JVM, etc.) to be initiated. Mobile code installer **711** will also recognize sandbox package **340** as an executable and will attempt to initiate sandbox package **340** at its “beginning.” Protection engine **400** processing corresponding to destination **700** use of a such a loader, however, will have resulted in the “beginning” of sandbox package **340** as corresponding to the beginning of MPC **341**, as noted with regard to the above FIG. *4* example.

Such protection engine processing will therefore cause a mobile code installer (e.g. OS loader **711**, for clarity sake) to initiate MPC **341**. In other cases, other processing might also be utilized for causing such initiation or further protection system operation. Protection engine processing also enables MPC **341** to effectively form a protection “sandbox” around Downloadable (e.g. detected-Downloadable or “XEQ”) **343**, to monitor Downloadable **343**, intercept determinable Downloadable **343** operation (such as attempted accesses of Downloadable **343** to destination resources) and, if “malicious”, to cause one or more other operations to occur (e.g. providing an alert, offloading the Downloadable, offloading the MPC, providing only limited resource access, possibly in a particular address space or with regard to a particularly “safe” resource or resource operation, etc.).

MPC **341**, in the present OS example, executes MPC element installation and installs any policies, causing MPC **341** and protection policies **342** to be loaded into a first memory space, P1. MPC **341** then initiates loading of Downloadable **343**. Such Downloadable initiation causes OS loader **711** to load Downloadable **343** into a further working memory space-P2 **703** along with an API import table (“IAT”) **731** for providing Downloadable **631** with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9x and Windows NT), which can be accommodated by agent generator configurability, such as that given above. MPC **341** therefore has at least initial access to API IAT **731** of Downloadable **632**, and provides for diverting, evaluating and responding to attempts by Downloadable **632** to utilize system APIs **731**, or further in accordance with protection poli-

cies **342**. In addition to API diverting, MPC **341** can also install filter drivers, which can be used for controlling access to resources such as a Downloadable-destination file system or registry. Filter driver installation can be conducted as documented in the MSDN or using other suitable methods.

Turning to FIG. *8* with reference to FIG. *7b*, an MPC **341** according to an embodiment of the invention includes a package extractor **801**, executable installer **802**, sandbox engine (attempt) analyzer **805**, policy enforcer **806** and MPC de-installer **807**. Package extractor **801** is initiated upon initiation of MPC **341**, and extracts MPC **341** elements and protection policies **342**. Executable installer **802** further initiates installation of a Downloadable by extracting the downloadable from the protected package, and loading the process into memory in suspended mode (so it only loads into memory, but does not start to run). Such installation further causes the operating system to initialize the Downloadable’s IAT **731** in the memory space of the downloadable process, P2, as already noted.

Sandbox engine installer **803** (running in process space P1) then installs the sandbox engine (**803-805**) and policies **342** into the downloadable process space P2. This is done in different way in each operating system (e.g. see above). Resource access diverter **804** further modifies those Downloadable-API IAT entries that correspond with protection policies **342**, thereby causing corresponding Downloadable accesses via Downloadable-API IAT **731** to be diverted resource access analyzer **805**.

During Downloadable operation, resource access analyzer or “RAA” **805** receives and determines a response to diverted Downloadable (i.e. “malicious”) operations in accordance with corresponding protection policies of policies **342**. (RAA **805** or further elements, which are not shown, can further similarly provide for other security mechanisms that might also be implemented.) Malicious operations can for example include, in a Windows environment: file operations (e.g. reading, writing, deleting or renaming a file), network operations (e.g. listen on or connect to a socket, send/receive data or view intranet), OS registry or similar operations (read/write a registry item), OS operations (exit OS/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using “dummy” or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

The FIG. *9* flowchart illustrates a protection method according to an embodiment of the invention. In step **901**, a protection engine monitors the receipt, by a server or other re-communicator of information, and receives such information intended for a protected information-destination (i.e. a potential-Downloadable) in step **903**. Steps **905-911** depict an adjunct trustworthiness protection that can also be provided, wherein the protection engine determines whether the source of the received information is known to be “unfriendly” and, if so, prevents current (at least unaltered) delivery of the potential-Downloadable and provides any

suitable alerts. (The protection engine might also continue to perform Downloadable detection and nevertheless enable delivery or protected delivery of a non-Downloadable, or avoid detection if the source is found to be “trusted”, among other alternatives enabled by the teachings herein.)

If, in step **913**, the potential-Downloadable source is found to be of an unknown or otherwise suitably authenticated/certified source, then the protection engine determines whether the potential-Downloadable includes executable code in step **915**. If the potential-Downloadable does not include executable code, then the protection engine causes the potential-Downloadable to be delivered to the information-destination in its original form in step **917**, and the method ends. If instead the potential-Downloadable is found to include executable code in step **915** (and is thus a “detected-Downloadable”), then the protection engine forms a sandboxed package in step **919** and causes the protection agent to be delivered to the information-Destination in step **921**, and the method ends. As was discussed earlier, a suitable protection agent can include mobile protection code, policies and the detected-Downloadable (or information corresponding thereto).

The FIG. **10a** flowchart illustrates a method for analyzing a potential-Downloadable, according to an embodiment of the invention. As shown, one or more aspects can provide useful indicators of the inclusion of executable code within the potential-Downloadable. In step **1001**, the protection engine determines whether the potential-Downloadable indicates an executable file type, for example, by comparing one or more included file headers for file type indicators (e.g. extensions or other descriptors). The indicators can be compared against all known file types executable by all protected Downloadable destinations, a subset, in accordance with file types executable or desirably executable by the Downloadable-destination, in conjunction with a particular user, in conjunction with available information or operability at the destination, various combinations, etc.

Where content analysis is conducted, in step **1003** of FIG. **10a**, the protection engine analyzes the potential-Downloadable and determines in accordance therewith whether the potential-Downloadable does or is likely to include binary information, which typically indicates executable code. The protection engine further analyzes the potential-Downloadable for patterns indicative of included executable code in step **1003**. Finally, in step **1005**, the protection engine determines whether the results of steps **1001** and **1003** indicate that the potential-Downloadable more likely includes executable code (e.g. via weighted comparison of the results with a suitable level indicating the inclusion or exclusion of executable code). The protection engine, given a suitably high confidence indicator of the inclusion of executable code, treats the potential-Downloadable as a detected-Downloadable.

The FIG. **10b** flowchart illustrates a method for forming a sandboxed package according to an embodiment of the invention. As shown, in step **1011**, a protection engine retrieves protection parameters and forms mobile protection code according to the parameters. The protection engine further, in step **1013**, retrieves protection parameters and forms protection policies according to the parameters. Finally, in step **1015**, the protection engine couples the mobile protection code, protection policies and received-information to form a sandboxed package. For example, where a Downloadable-destination utilizes a standard windows executable, coupling can further be accomplished via concatenating the MPC for delivery of MPC first, policies second, and received information third. (The protection parameters can, for example,

include parameters relating to one or more of the Downloadable destination device/process, user, supervisory constraints or other parameters.)

The FIG. **11** flowchart illustrates how a protection method performed by mobile protection code (“MPC”) according to an embodiment of the invention includes the MPC installing MPC elements and policies within a destination device in step **1101**. In step **1102**, the MPC loads the Downloadable without actually initiating it (i.e. for executables, it will start a process in suspended mode). The MPC further forms an access monitor or “interceptor” for monitoring or “intercepting” downloadable destination device access attempts within the destination device (according to the protection policies in step **1103**, and initiates a corresponding Downloadable within the destination device in step **1105**.

If, in step **1107**, the MPC determines, from monitored/intercepted information, that the Downloadable is attempting or has attempted a destination device access considered undesirable or otherwise malicious, then the MPC performs steps **1109** and **1111**; otherwise the MPC returns to step **1107**. In step **1109**, the MPC determines protection policies in accordance with the access attempt by the Downloadable, and in step **1111**, the MPC executes the protection policies. (Protection policies can, for example, be retrieved from a temporary, e.g. memory/cache, or more persistent storage.)

As shown in the FIG. **12a** example, the MPC can provide for intercepting Downloadable access attempts by a Downloadable by installing the Downloadable (but not executing it) in step **1201**. Such installation will cause a Downloadable executor, such as a the Windows operating system, to provide all required interfaces and parameters (such as the IAT, process ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **12b** example further illustrates an example of how the MPC can apply suitable policies in accordance with an access attempt by a Downloadable. As shown, the MPC receives the Downloadable access request via the modified IAT in step **1211**. The MPC further queries stored policies to determine a policy corresponding to the Downloadable access request in step **1213**.

The foregoing description of preferred embodiments of the invention is provided by way of example to enable a person skilled in the art to make and use the invention, and in the context of particular applications and requirements thereof. Various modifications to the embodiments will be readily apparent to those skilled in the art, and the generic principles defined herein may be applied to other embodiments and applications without departing from the spirit and scope of the invention. Thus, the present invention is not intended to be limited to the embodiments shown, but is to be accorded the widest scope consistent with the principles, features and teachings disclosed herein. The embodiments described herein are not intended to be exhaustive or limiting. The present invention is limited only by the following claims.

What is claimed is:

1. A computer-based method, comprising the steps of:
  - receiving an incoming Downloadable;
  - performing a hashing function on the incoming Downloadable to compute an incoming Downloadable ID;
  - retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on the incoming Downloadable ID, the security profile data

US 7,613,926 B2

21

including a list of suspicious computer operations that may be attempted by the Downloadable;  
 appending a representation of the retrieved Downloadable security profile data to the incoming Downloadable, to generate an appended Downloadable; and  
 transmitting the appended Downloadable to a destination computer.

2. The computer-based method of claim 1 wherein the Downloadable includes an applet.

3. The computer-based method of claim 1 wherein the Downloadable includes an active control.

4. The computer-based method of claim 1 wherein the Downloadable includes program script.

5. The computer-based method of claim 1 wherein suspicious computer operations include calls made to an operating system, a file system, a network system, and to memory.

6. The computer-based method of claim 1 wherein the Downloadable security profile data includes a URL from where the Downloadable originated.

7. The computer-based method of claim 1 wherein the Downloadable security profile data includes a digital certificate.

8. A system for managing Downloadables, comprising:  
 a receiver for receiving an incoming Downloadable;  
 a Downloader identifier for performing a hashing function on the incoming Downloadable to compute an incoming Downloadable ID;

a database manager for retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on the incoming Downloadable ID, the security profile data including a list of suspicious computer operations that may be attempted by the Downloadable;

a file appender coupled with said receiver for appending are presentation of the Downloadable security profile data to the incoming Downloadable, to generate an appended Downloadable; and

a transmitter coupled with said file appender, for transmitting the appended Downloadable to a destination computer.

9. The system of claim 8 wherein the Downloadable includes an applet.

10. The system of claim 8 wherein the Downloadable includes an active control.

11. The system of claim 8 wherein the Downloadable includes program script.

12. The system of claim 8 wherein suspicious computer operations include calls made to an operating system, a file system, a network system, and to memory.

13. The system of claim 8 wherein the Downloadable security profile data includes a URL from where the Downloadable originated.

14. The system of claim 8 wherein the Downloadable security profile data includes a digital certificate.

15. A computer-based method, comprising the steps of:  
 receiving an incoming Downloadable;

performing a hashing function on the incoming Downloadable to compute an incoming Downloadable ID;

retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on the incoming Downloadable ID, the security profile data including a list of suspicious computer operations that may be attempted by the Downloadable; and

22

transmitting the incoming Downloadable and a representation of the retrieved Downloadable security profile data to a destination computer, via a transport protocol transmission.

16. The computer-based method of claim 15 wherein the Downloadable includes an applet.

17. The computer-based method of claim 15 wherein the Downloadable includes an active control.

18. The computer-based method of claim 15 wherein the Downloadable includes program script.

19. The computer-based method of claim 15 wherein suspicious computer operations include calls made to an operating system, a file system, a network system, and to memory.

20. The computer-based method of claim 15 wherein the Downloadable security profile data includes a URL from where the Downloadable originated.

21. The computer-based method of claim 15 wherein the Downloadable security profile data includes a digital certificate.

22. A system for managing Downloadables, comprising:  
 a receiver for receiving an incoming Downloadable;  
 a Downloadable identifier for performing a hashing function on the incoming Downloadable to compute an incoming Downloadable ID;

a database manager for retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on the incoming Downloadable ID, the security profile data including a list of suspicious computer operations that may be attempted by the Downloadable; and

a transmitter coupled with said receiver, for transmitting the incoming Downloadable and a representation of the retrieved Downloadable security profile data to a destination computer, via a transport protocol transmission.

23. The system of claim 22 wherein the Downloadable includes an applet.

24. The system of claim 22 wherein the Downloadable includes an active control.

25. The system of claim 22 wherein the Downloadable includes program script.

26. The system of claim 22 wherein suspicious computer operations include calls made to an operating system, a file system, a network system, and to memory.

27. The system of claim 22 wherein the Downloadable security profile data includes a URL from where the Downloadable originated.

28. The system of claim 22 wherein the Downloadable security profile data includes a digital certificate.

29. A computer-readable storage medium storing program code for causing at least one computing device to:  
 receive an incoming Downloadable;

perform a hashing function on the incoming Downloadable to compute an incoming Downloadable ID;

retrieve security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on the incoming Downloadable ID, the security profile data including a list of suspicious computer operations that may be attempted by the Downloadable;

append a representation of the retrieved Downloadable security profile data to the incoming Downloadable, to generate an appended Downloadable; and  
 transmit the appended Downloadable to a destination computer.

30. A computer-readable storage medium storing program code for causing at least one computing device to:



US 7,613,926 B2

23

receive an incoming Downloadable;  
perform a hashing function on the incoming Downloadable  
to compute an incoming Downloadable ID;  
retrieve security profile data for the incoming Download-  
able from a database of Downloadable security profiles  
indexed according to Downloadable IDs, based on the  
incoming Downloadable ID, the security profile data

24

including a list of suspicious computer operations that  
may be attempted by the Downloadable; and  
transmit the incoming Downloadable and a representation  
of the retrieved Downloadable security profile data to a  
destination computer, via a transport protocol transmis-  
sion.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 7,613,926 B2  
APPLICATION NO. : 11/370114  
DATED : November 3, 2009  
INVENTOR(S) : Edery et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

In Item (63), "Related U.S. Application Data," please add the following information to the end of the information within Item (63):

-- which is a continuation of application No. 08/790,097, filed January 29, 1997, now U.S. Patent No. 6,167,520. --

In Column 1, please replace the paragraph following the paragraph title "Priority Reference To Related Applications" with the following paragraph:

-- This application is a continuation of assignee's application Ser. No. 09/861,229, filed on May 17, 2001, now U.S. Pat. No. 7,058,822, entitled "Malicious Mobile Code Runtime Monitoring System And Methods", which is hereby incorporated by reference. U.S. application Ser. No. 09/861,229 claims benefit of provisional application Ser. No. 60/205,591, entitled "Computer Network Malicious Code Run-time Monitoring," filed on May 17, 2000 by inventors Nimrod Itzhak Vered, et al., which is hereby incorporated by reference. U.S. application Ser. No. 09/861,229 is also a Continuation-In-Part of U.S. patent application Ser. No. 09/539,667, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul, now U.S. Pat. No. 6,804,780, and hereby incorporated by reference, which is a continuation of assignee's patent application U.S. Ser. No. 08/964,388, filed on Nov. 6, 1997, now U.S. Pat. No. 6,092,194, also entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables" and hereby incorporated by reference. U.S. Ser. No. 09/861,229 is also a Continuation-In-Part of U.S. patent application Ser. No. 09/551,302, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul, now U.S. Pat. No. 6,480,962, which is hereby incorporated by reference, which is a continuation of U.S. application Ser. No. 08/790,097, now U.S. Patent No. 6,167,520 entitled "System and Method For Protecting a Client From Hostile Downloadables", filed January 29, 1997 by inventor Shlomo Touboul. --

Signed and Sealed this  
Twenty-fifth Day of July, 2017



Joseph Matal  
*Performing the Functions and Duties of the  
Under Secretary of Commerce for Intellectual Property and  
Director of the United States Patent and Trademark Office*

# **EXHIBIT 3**



US006804780B1

(12) **United States Patent**  
Touboul

(10) **Patent No.:** US **6,804,780 B1**  
(45) **Date of Patent:** \***Oct. 12, 2004**

(54) **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

(75) Inventor: **Shlomo Touboul, Kefar-haim (IL)**

(73) Assignee: **Finjan Software, Ltd., Netanya (IL)**

(\*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

This patent is subject to a terminal disclaimer.

(21) Appl. No.: **09/539,667**

(22) Filed: **Mar. 30, 2000**

**Related U.S. Application Data**

- (63) Continuation of application No. 08/964,388, filed on Nov. 6, 1997, now Pat. No. 6,092,194.
- (60) Provisional application No. 60/030,639, filed on Nov. 8, 1996.
- (51) **Int. Cl.**<sup>7</sup> ..... **H04L 9/00; G06F 11/30**
- (52) **U.S. Cl.** ..... **713/181; 713/201; 713/176; 717/178**
- (58) **Field of Search** ..... **713/200, 201, 713/176, 181; 709/223, 225, 227, 229; 717/168-178**

(56) **References Cited**

**U.S. PATENT DOCUMENTS**

5,077,677 A	12/1991	Murphy et al.
5,359,659 A	10/1994	Rosenthal
5,361,359 A	11/1994	Tajalli et al.
5,485,409 A	1/1996	Gupta et al.
5,485,575 A	1/1996	Chess et al.

5,572,643 A	11/1996	Judson
5,579,509 A *	11/1996	Furtney et al. .... 703/27
5,606,668 A	2/1997	Shwed
5,623,600 A	4/1997	Ji et al.
5,638,446 A	6/1997	Rubin
5,692,047 A	11/1997	McManis
5,692,124 A	11/1997	Holden et al.
5,720,033 A	2/1998	Deo
5,724,425 A	3/1998	Chang et al.
5,740,248 A	4/1998	Fieres et al.
5,761,421 A	6/1998	van Hoff et al.

(List continued on next page.)

**FOREIGN PATENT DOCUMENTS**

EP	1091276 A1 *	4/2001	.....	G06F/1/00
EP	1132796 A1 *	9/2001	.....	G06F/1/00

**OTHER PUBLICATIONS**

Khare, "Microsoft Authenticode Analyzed" Jul. 22, 1996, xent.com/ForK-archiv/summer96/0338.html, p. 1-2.\*

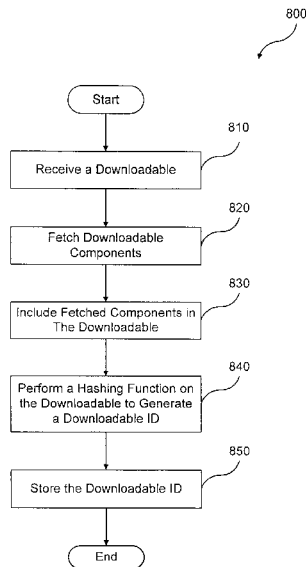
(List continued on next page.)

*Primary Examiner*—Ayaz Sheikh  
*Assistant Examiner*—Christopher Revak  
(74) *Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

(57) **ABSTRACT**

A computer-based method for generating a Downloadable ID to identify a Downloadable, including obtaining a Downloadable that includes one or more references to software components required by the Downloadable, fetching at least one software component identified by the one or more references, and performing a function on the Downloadable and the fetched software components to generate a Downloadable ID. A system and a computer-readable storage medium are also described and claimed.

**18 Claims, 10 Drawing Sheets**



## US 6,804,780 B1

Page 2

## U.S. PATENT DOCUMENTS

5,765,205	A	6/1998	Breslau et al.	
5,784,459	A	7/1998	Devarakonda et al.	
5,796,952	A	8/1998	Davis et al.	
5,805,829	A	9/1998	Cohen et al.	
5,832,208	A	11/1998	Chen et al.	
5,832,274	A *	11/1998	Cutler et al. ....	717/171
5,850,559	A	12/1998	Angelo et al.	
5,859,966	A	1/1999	Hayman et al.	
5,864,683	A	1/1999	Boebert et al.	
5,892,904	A	4/1999	Atkinson et al.	
5,951,698	A	9/1999	Chen et al.	
5,956,481	A	9/1999	Walsh et al.	
5,974,549	A	10/1999	Golan	
5,978,484	A *	11/1999	Apperson et al. ....	705/54
5,983,348	A	11/1999	Ji	
6,092,194	A *	7/2000	Touboul .....	713/200
6,154,844	A *	11/2000	Touboul et al. ....	713/201
6,339,829	B1 *	1/2002	Beadle et al. ....	713/201

## OTHER PUBLICATIONS

“Release Notes for the Microsoft ActiveX Development Kit”, Aug. 13, 1996, [activex.adsp.or.jp/inetsdk/readme.txt](http://activex.adsp.or.jp/inetsdk/readme.txt), p. 1–10.\*

“Microsoft ActiveX Software Development Kit” Aug. 12, 1996, [activex.adsp.or.jp/inetsdk/help/overview.htm](http://activex.adsp.or.jp/inetsdk/help/overview.htm), p. 1–6.\*

Doyle et al, “Microsoft Press Computer Dictionary” 1993, Microsoft Press, 2nd Edition, p. 137–138.\*

Schmitt, “.EXE. files, OS–2 style” Nov. 1988, PC Tech Journal via dialog search, vol. 6, #11, p. 76–78.\*

Jim K. Omura, “Novel Applications of Cryptography in Digital Communications”, IEEE Communications Magazine, May, 1990; pp. 21–29.

Okamoto, E. et al., “ID–Based Authentication System For Computer Virus Detection”, IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013/5194, Jul. 19, 1990, Abstract and pp. 1169–1170. URL: <http://iel.ihs.com:80/cgi-bin/iel.cgi?se2ehts%26ViewTemplate%3ddocview%5fb%2ehts>.

IBM AntiVirus User’s Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, pp. 6–7.

Norvin Leach et al, “IE 3.0 Applets Will Earn Certification”, PC Week, vol. 13, No. 29, Jul. 22, 1996, 2 pages.

“Finjan Software Releases SurfinBoard, Industry’s First JAVA Security Product For the World Wide Web”, Article published on the Internet by Finjan Software Ltd., Jul. 29, 1996, 1 page.

“Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™” Article published on the Internet by Finjan Software Ltd., 1996, 2 Pages.

Microsoft® Authenticode Technology, “Ensuring Accountability and Authenticity for Software Components on the Internet”, Microsoft Corporation, Oct. 1996, including Abstract, Contents, Introduction and pp. 1–10.

“Finjan Announces a Personal Java™ Firewall For Web Browsers—the SurfinShield™ 1.6 (formerly known as SurfinBoard)”, Press Release of Finjan Releases SurfinShield 1.6, Oct. 21, 1996, 2 pages.

Company Profile “Finjan—Safe Surfing, The Java Security Solutions Provider”, Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

“Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0” Las Vegas Convention Center/Pavilion 5 P5551, Nov. 18, 1996, 3 pages.

“Java Security: Issues & Solutions” Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

“Products” Article published on the Internet, 7 pages.

Mark LaDue, “Online Business Consultant: Java Security: Whose Business Is It?” Article published on the Internet, Home Page Press, Inc. 1996, 4 pages.

Web Page Article “Frequently Asked Questions About Authenticode”, Microsoft Corporation, last updated Feb. 17, 1997, Printed Dec. 23, 1998. URL: <http://www.microsoft.com/workshop/security/authcode/signfaq.asp#9>, pp. 1–13.

Zhang, X.N., “Secure Code Distribution”, IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, Jun., 1997, pp. 76–79.

\* cited by examiner

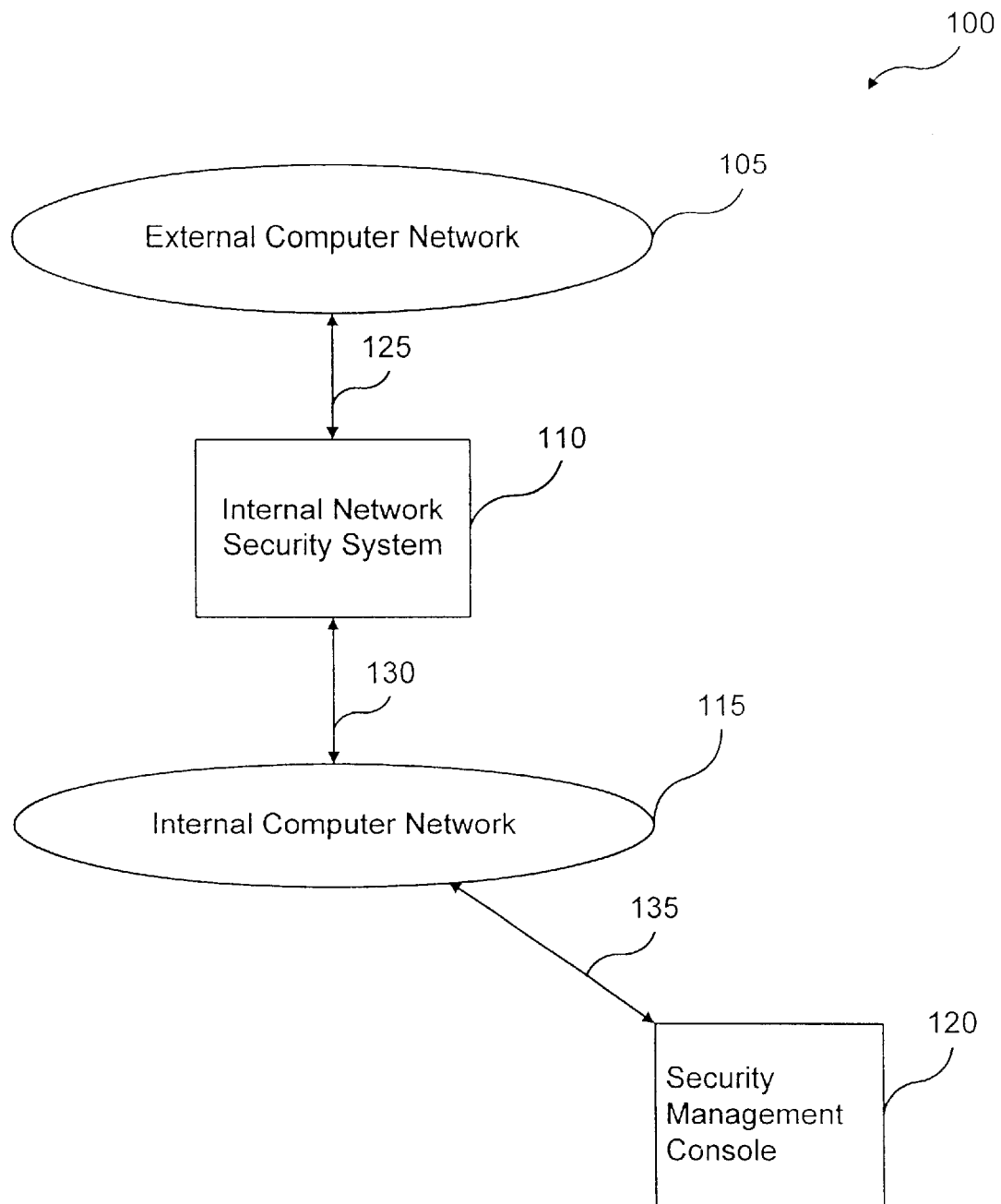


FIG. 1

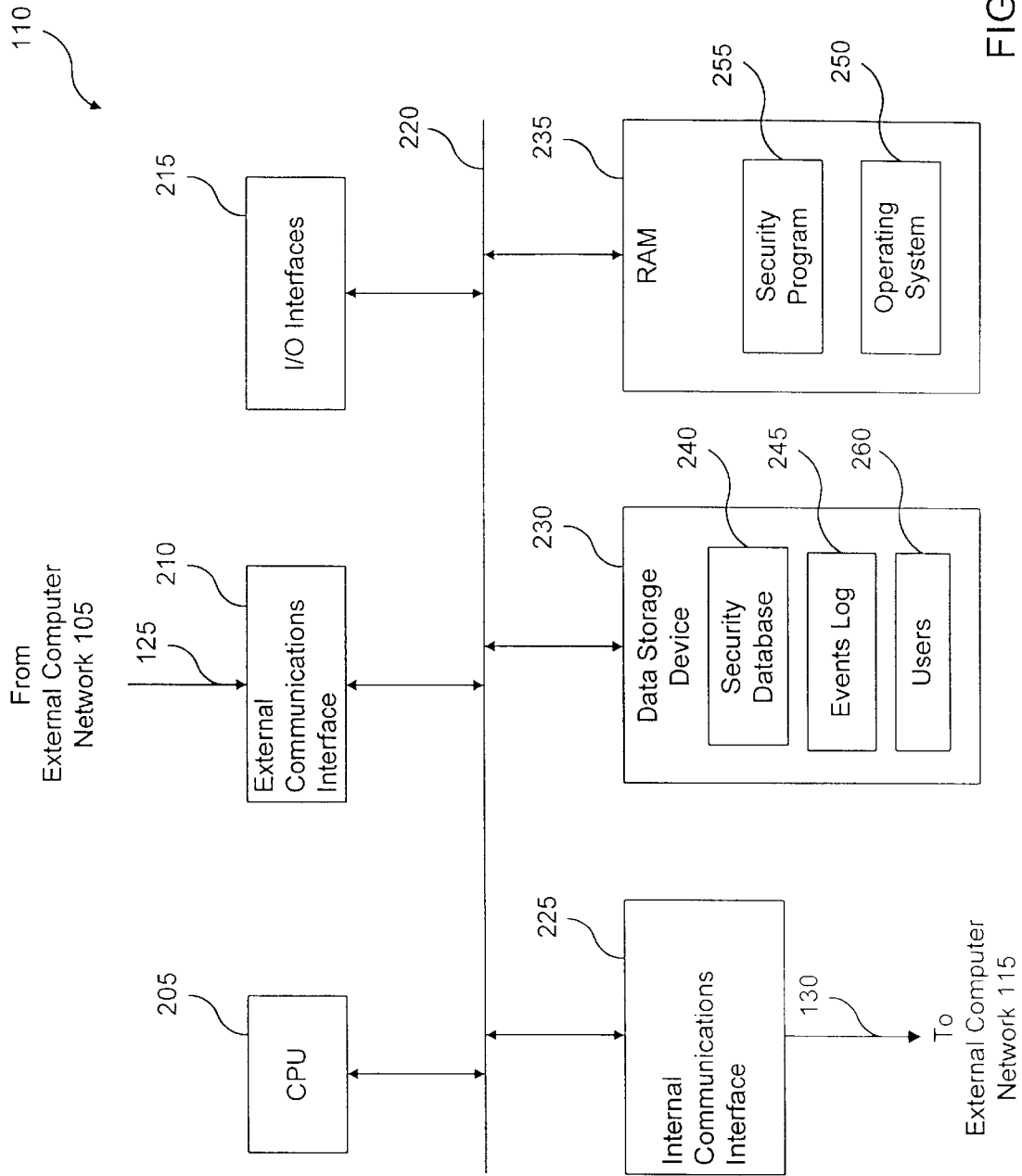


FIG. 2

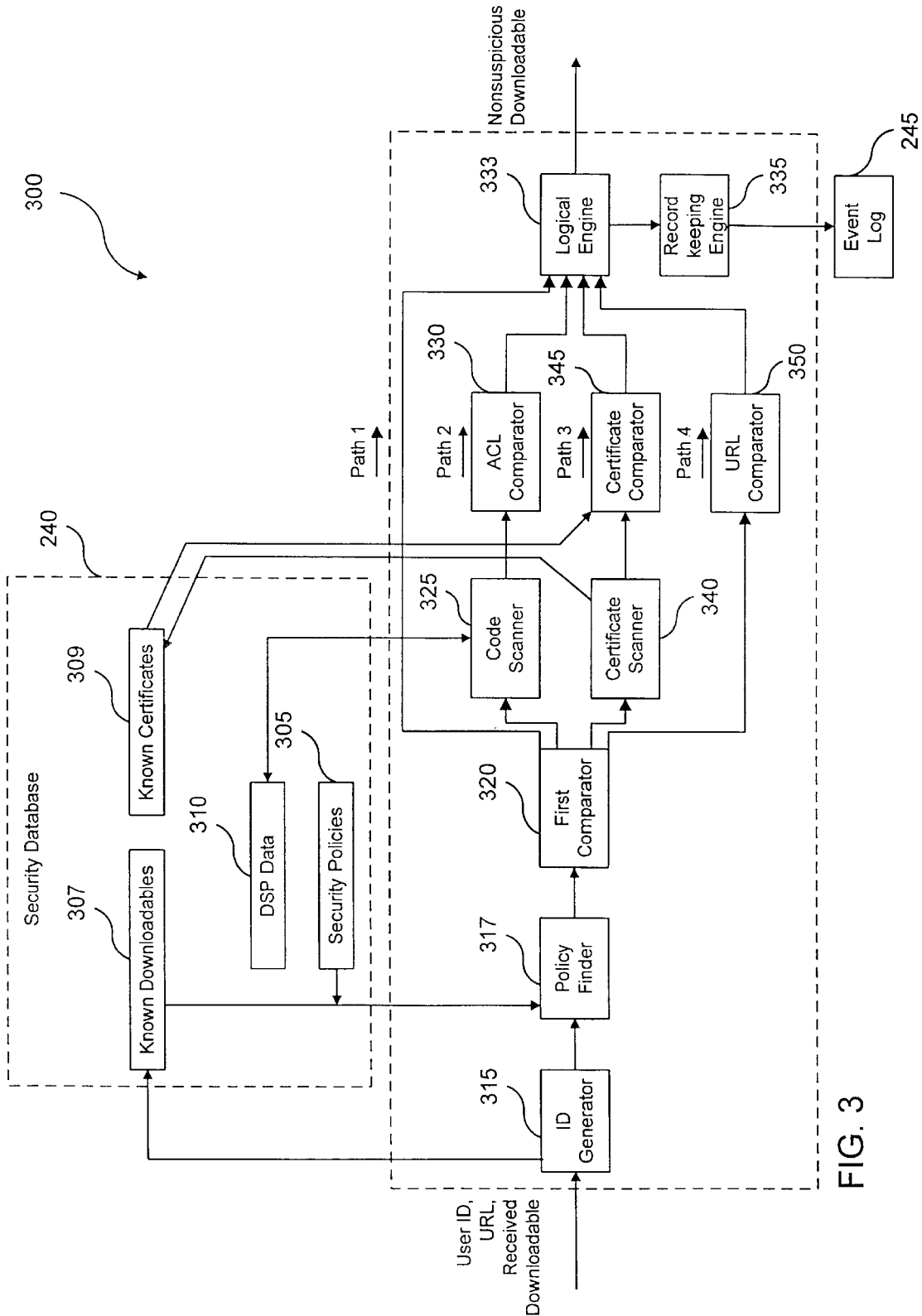


FIG. 3



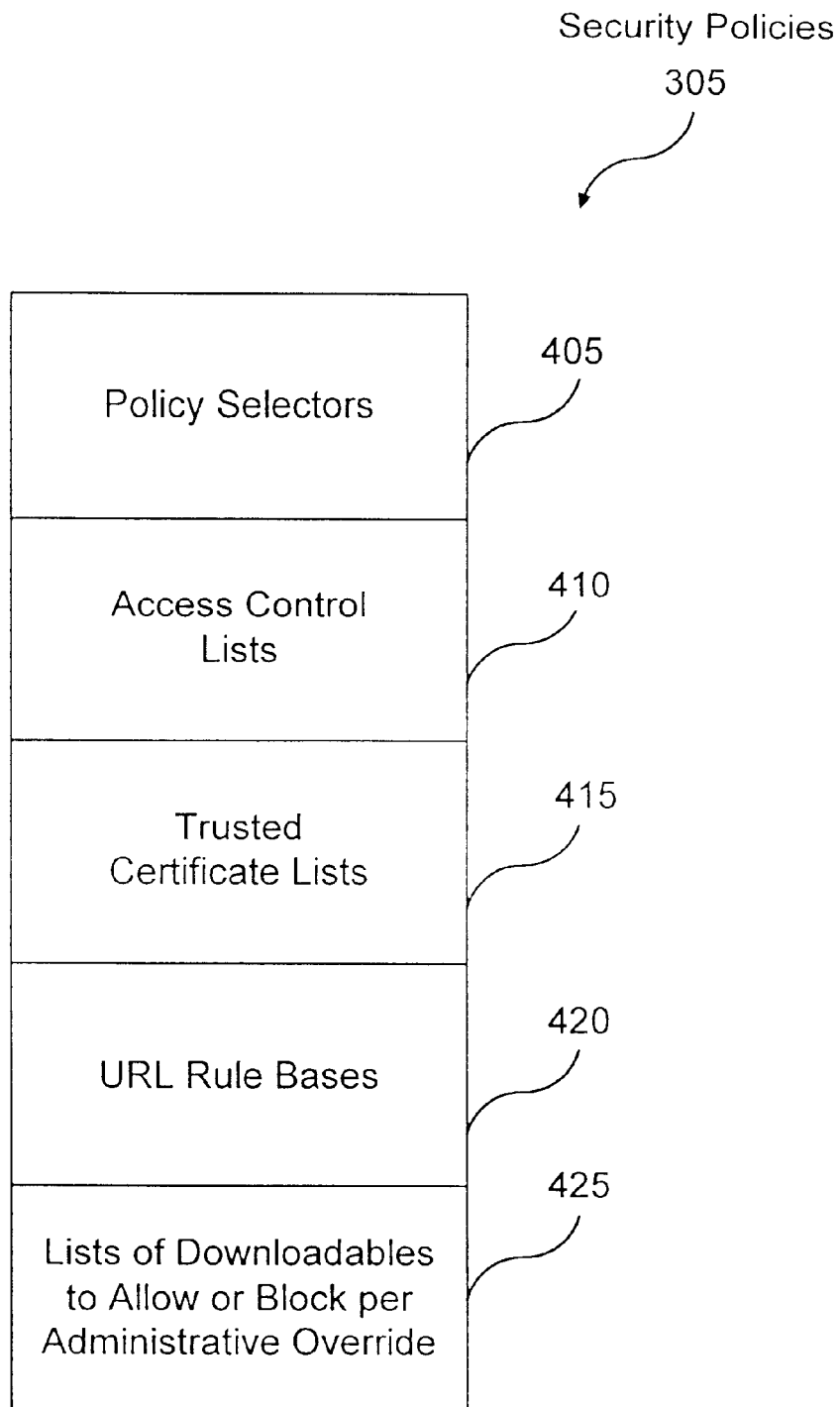


FIG. 4

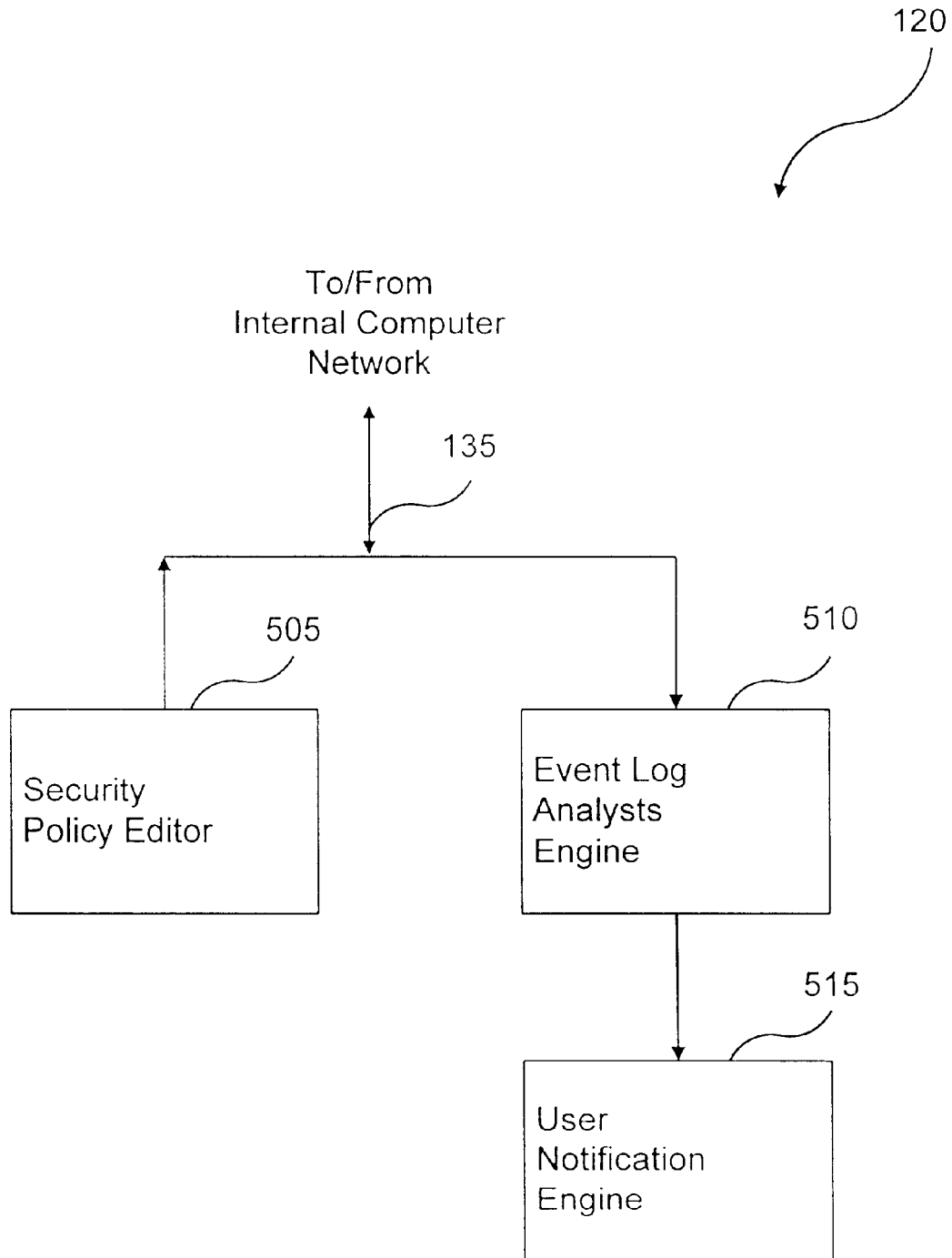


FIG. 5

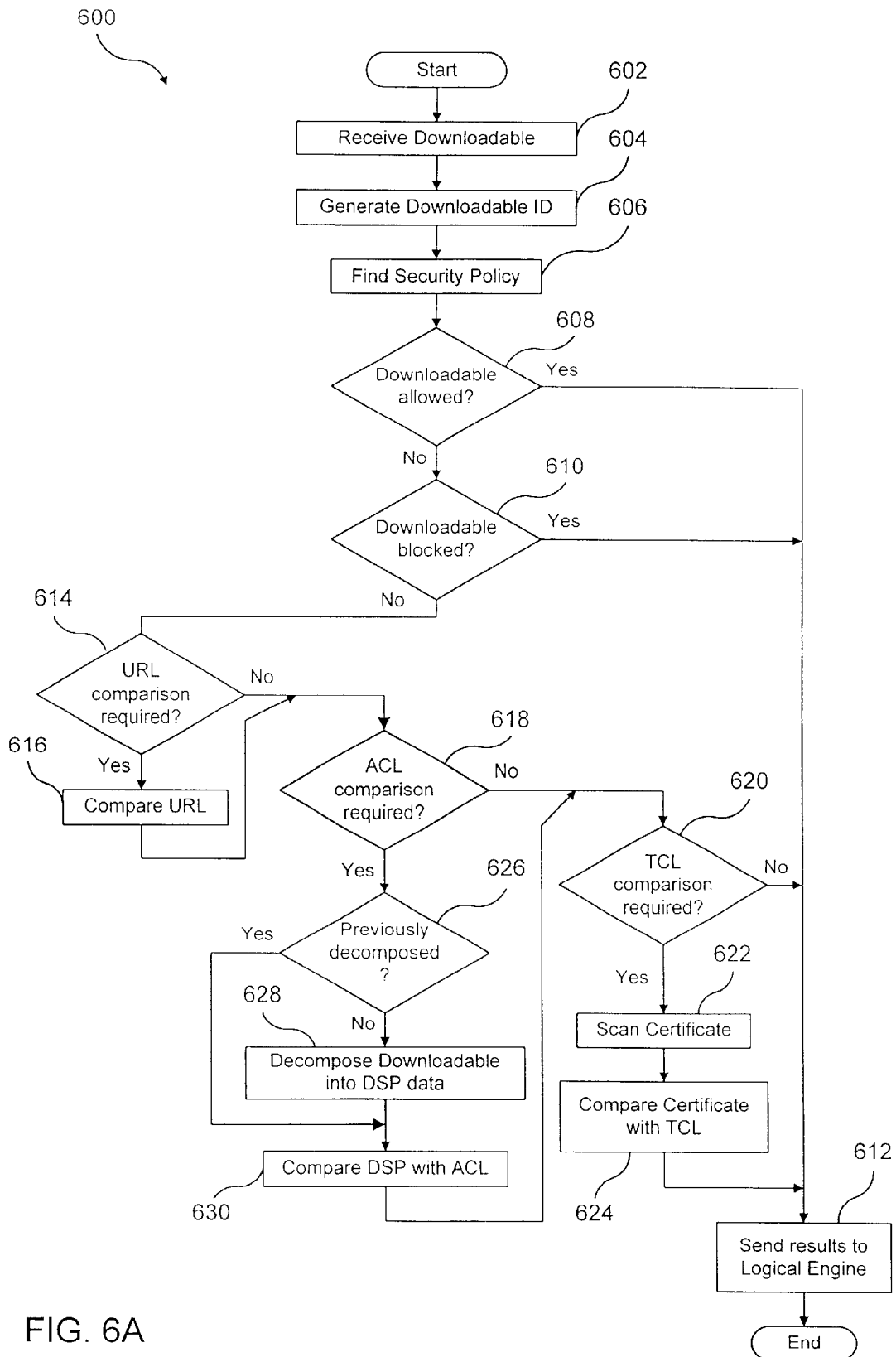


FIG. 6A

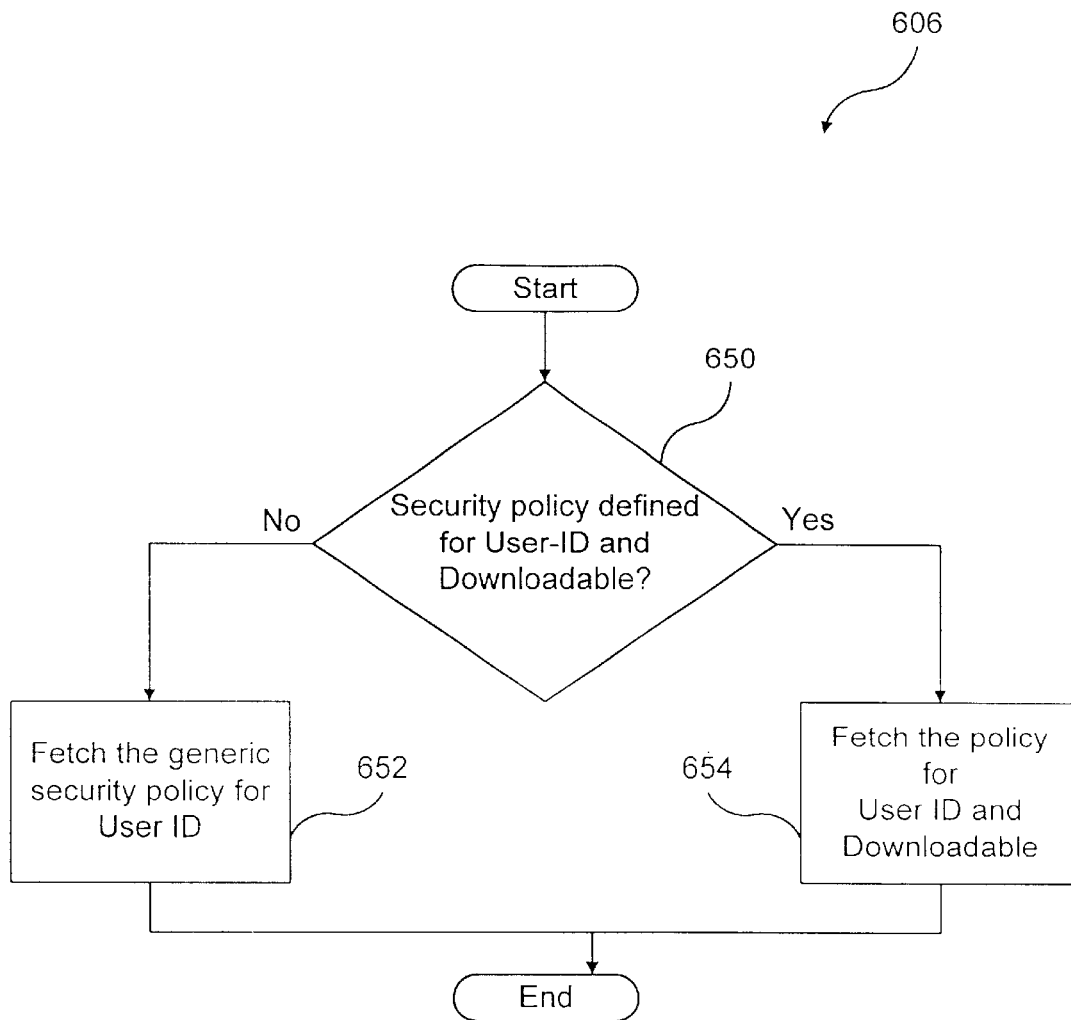


FIG. 6B

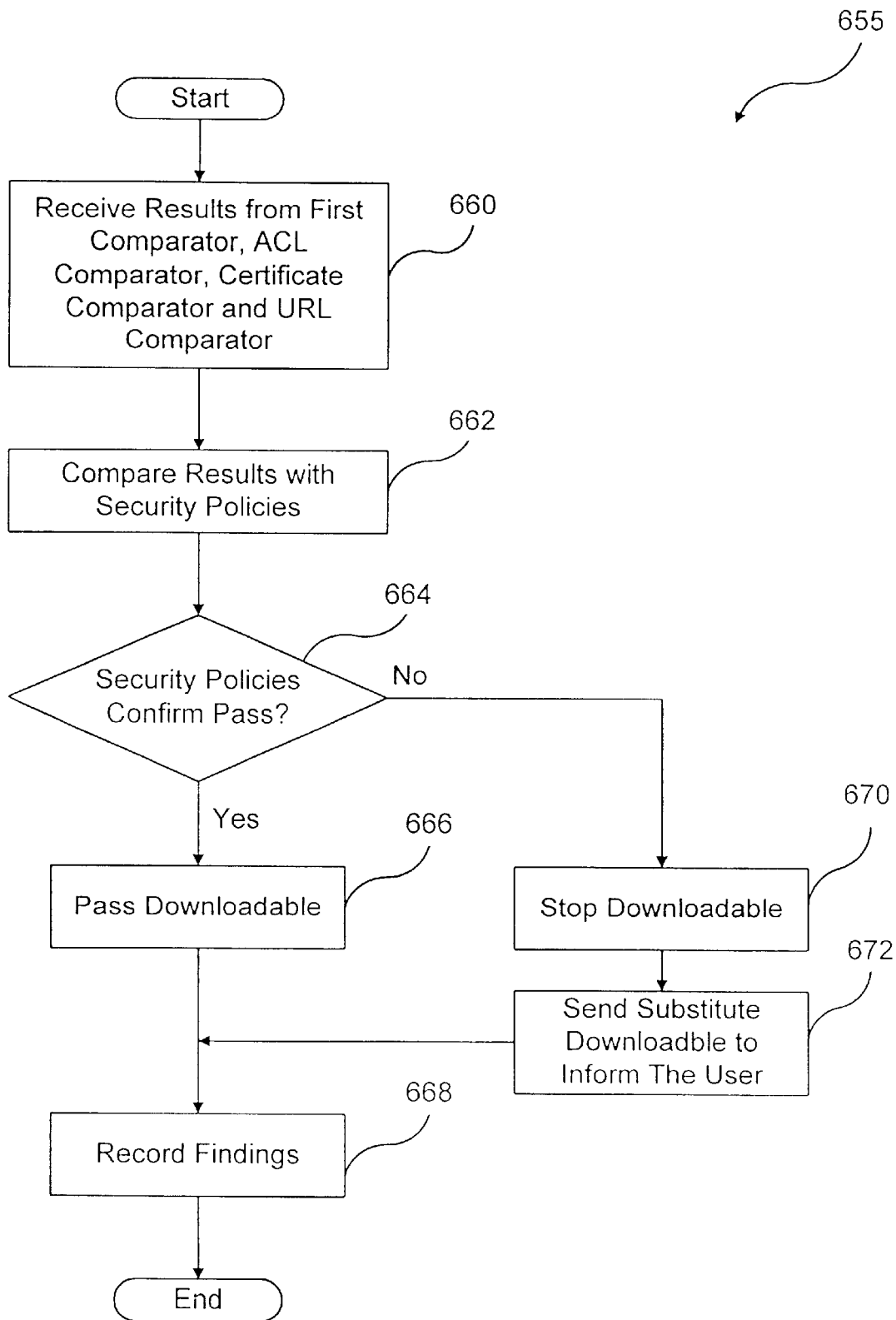


FIG. 6C

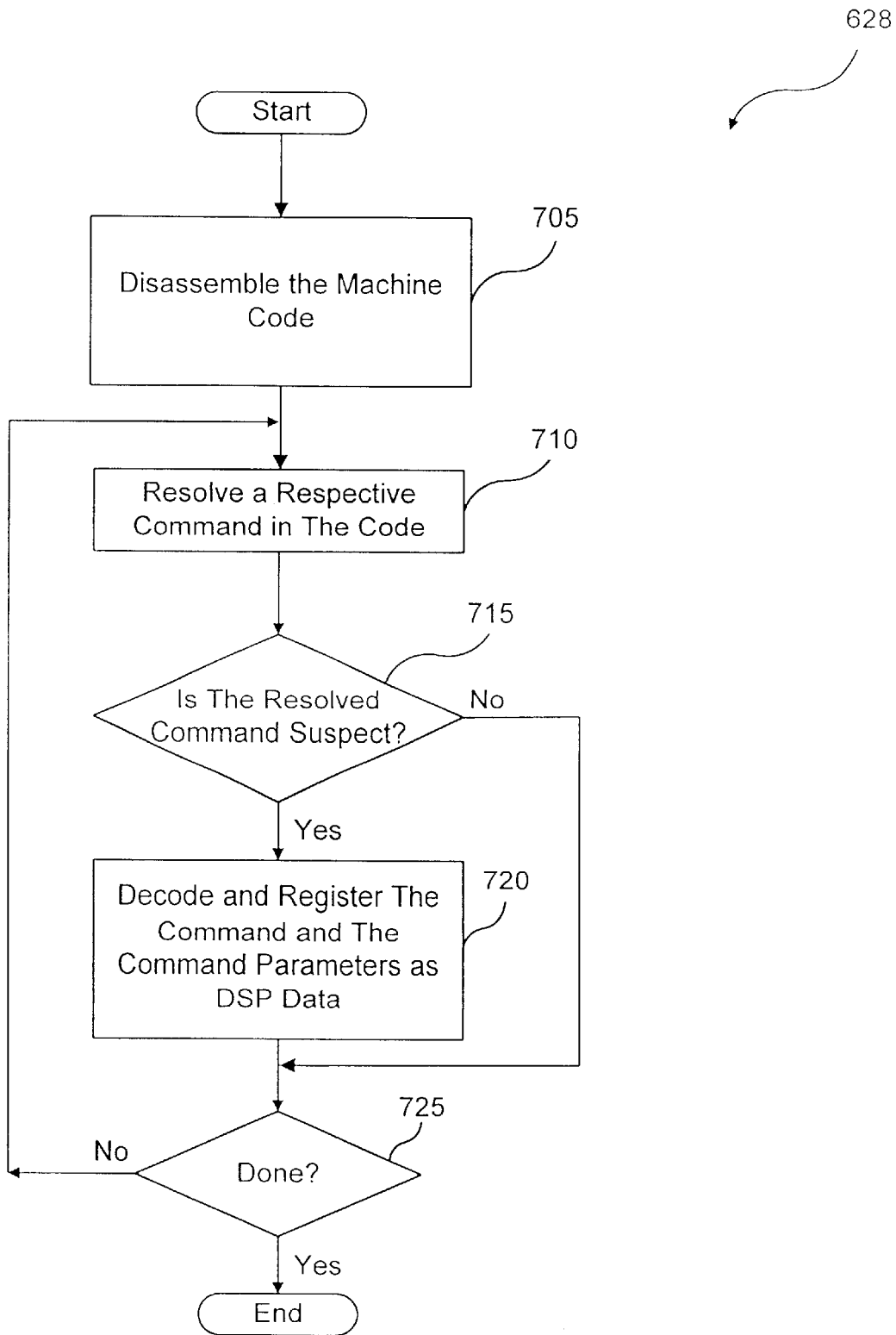


FIG. 7

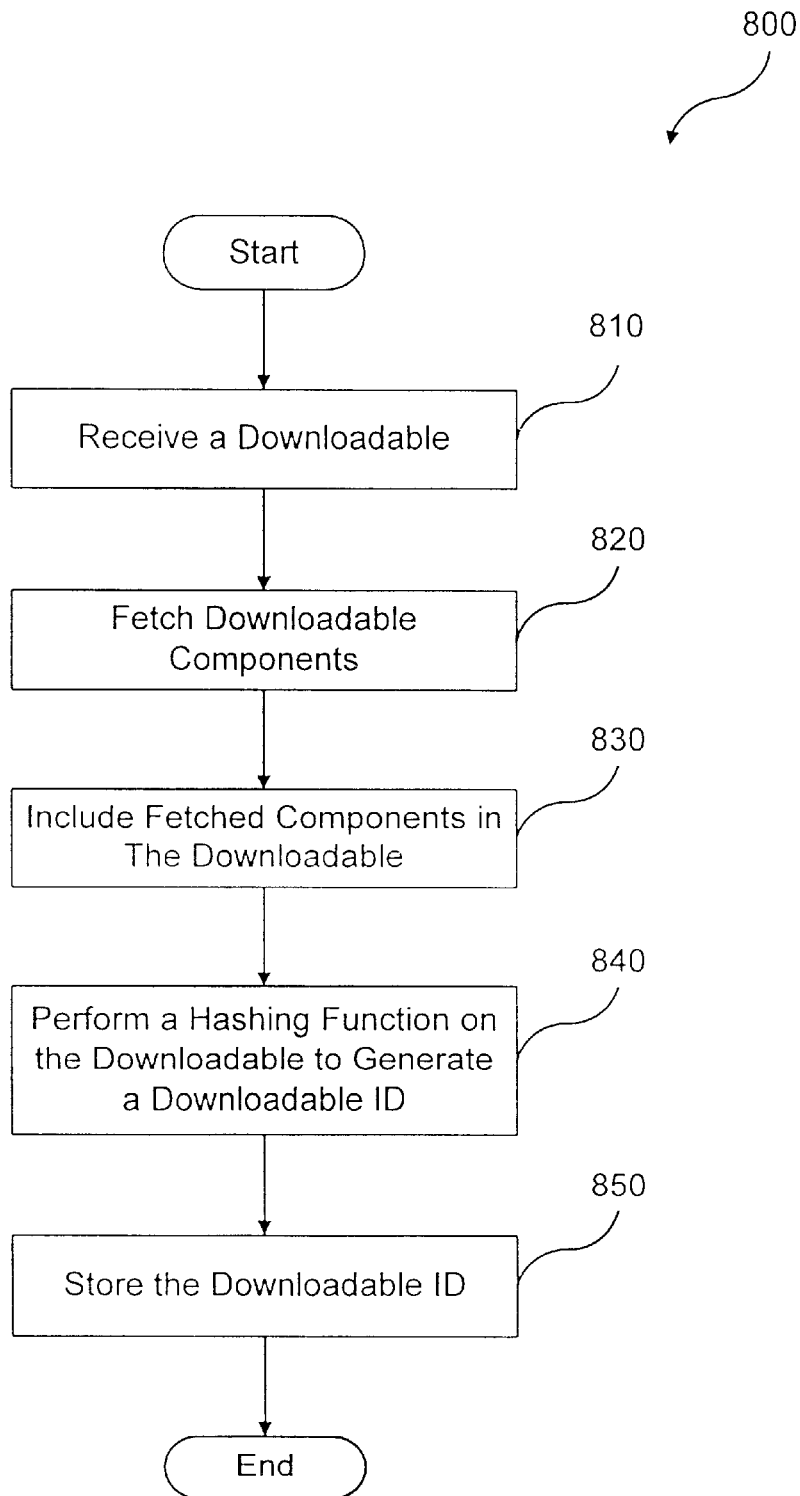


FIG. 8

US 6,804,780 B1

1

## SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES

### PRIORITY REFERENCE TO RELATED APPLICATION

This application is a continuation of and hereby incorporates by reference U.S. patent application Ser. No. 08/964, 388, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables," filed Nov. 6, 1997, which is now U.S. Pat. No. 6,092,194, which claims priority to provisional application Serial No. 60/030, 639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

### INCORPORATION BY REFERENCE TO RELATED APPLICATIONS

This application hereby incorporates by reference related U.S. patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, which is now U.S. Pat. No. 6,167,520, by inventor Shlomo Touboul; and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul.

### BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for protecting a computer and a network from hostile Downloadables.

#### 2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. Downloadable is typically requested by an ongoing process such as by an Internet browser or web engine. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Therefore, a system and method are needed to protect a network from hostile Downloadables.

### SUMMARY OF THE INVENTION

The present invention provides a system for protecting a network from suspicious Downloadables. The system com-

2

prises a security policy, an interface for receiving a Downloadable, and a comparator, coupled to the interface, for applying the security policy to the Downloadable to determine if the security policy has been violated. The Downloadable may include a Java™ applet, an ActiveX™ control, a JavaScript™ script, or a Visual Basic script. The security policy may include a default security policy to be applied regardless of the client to whom the Downloadable is addressed, a specific security policy to be applied based on the client or the group to which the client belongs, or a specific policy to be applied based on the client/group and on the particular Downloadable received. The system uses an ID generator to compute a Downloadable ID identifying the Downloadable, preferably, by fetching all components of the Downloadable and performing a hashing function on the Downloadable including the fetched components.

Further, the security policy may indicate several tests to perform, including (1) a comparison with known hostile and non-hostile Downloadables; (2) a comparison with Downloadables to be blocked or allowed per administrative override; (3) a comparison of the Downloadable security profile data against access control lists; (4) a comparison of a certificate embodied in the Downloadable against trusted certificates; and (5) a comparison of the URL from which the Downloadable originated against trusted and untrusted URLs. Based on these tests, a logical engine can determine whether to allow or block the Downloadable.

The present invention further provides a method for protecting a computer from suspicious Downloadables. The method comprises the steps of receiving a Downloadable, comparing the Downloadable against a security policy to determine if the security policy has been violated, and discarding the Downloadable if the security policy has been violated.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any suspicious operations, and thus may allow or block the Downloadable accordingly.

### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system, in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of the internal network security system of FIG. 1;

FIG. 3 is a block diagram illustrating details of the security program and the security database of FIG. 2;

FIG. 4 is a block diagram illustrating details of the security policies of FIG. 3;

FIG. 5 is a block diagram illustrating details of the security management console of FIG. 1;

FIG. 6A is a flowchart illustrating a method of examining for suspicious Downloadables, in accordance with the present invention;

FIG. 6B is a flowchart illustrating details of the step for finding the appropriate security policy of FIG. 6A;

FIG. 6C is a flowchart illustrating a method for determining whether an incoming Downloadable is to be deemed suspicious;

FIG. 7 is a flowchart illustrating details of the FIG. 6 step of decomposing a Downloadable; and



US 6,804,780 B1

3

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable.

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a network system 100, in accordance with the present invention. The network system 100 includes an external computer network 105, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a communications channel 125 to an internal network security system 110. The network system 100 further includes an internal computer network 115, such as a corporate Local Area Network (LAN), coupled via a communications channel 130 to the internal network computer system 110 and coupled via a communications channel 135 to a security management console 120.

The internal network security system 110 examines Downloadables received from external computer network 105, and prevents Downloadables deemed suspicious from reaching the internal computer network 115. It will be further appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of an internal computer network 115 component. It is to be understood that the term “suspicious” includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Security management console 120 enables viewing, modification and configuration of the internal network security system 110.

FIG. 2 is a block diagram illustrating details of the internal network security system 110, which includes a Central Processing Unit (CPU) 205, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 220. The internal network security system 110 further includes an external communications interface 210 coupled between the communications channel 125 and the signal bus 220 for receiving Downloadables from external computer network 105, and an internal communications interface 225 coupled between the signal bus 220 and the communications channel 130 for forwarding Downloadables not deemed suspicious to the internal computer network 115. The external communications interface 210 and the internal communications interface 225 may be functional components of an integral communications interface (not shown) for both receiving Downloadables from the external computer network 105 and forwarding Downloadables to the internal computer network 115.

Internal network security system 110 further includes Input/Output (I/O) interfaces 215 (such as a keyboard, mouse and Cathode Ray Tube (CRT) display), a data storage device 230 such as a magnetic disk, and a Random-Access Memory (RAM) 235, each coupled to the signal bus 220. The data storage device 230 stores a security database 240, which includes security information for determining whether a received Downloadable is to be deemed suspicious. The data storage device 230 further stores a users list 260 identifying the users within the internal computer network 115 who may receive Downloadables, and an event log 245 which includes determination results for each Downloadable examined and runtime indications of the internal network security system 110. An operating system 250 controls processing by CPU 205, and is typically stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution. A security program 255 controls

4

examination of incoming Downloadables, and also may be stored in data storage device 230 and loaded into RAM 235 (as illustrated) for execution by CPU 205.

FIG. 3 is a block diagram illustrating details of the security program 255 and the security database 240. The security program 255 includes an ID generator 315, a policy finder 317 coupled to the ID generator 315, and a first comparator 320 coupled to the policy finder 317. The first comparator 320 is coupled to a logical engine 333 via four separate paths, namely, via Path 1, via Path 2, via Path 3 and via Path 4. Path 1 includes a direct connection from the first comparator 320 to the logical engine 333. Path 2 includes a code scanner coupled to the first comparator 320, and an Access Control List (ACL) comparator 330 coupling the code scanner 325 to the logical engine 333. Path 3 includes a certificate scanner 340 coupled to the first comparator 320, and a certificate comparator 345 coupling the certificate scanner 340 to the logical engine 333. Path 4 includes a Uniform Resource Locator (URL) comparator 350 coupling the first comparator 320 to the logical engine 3330. A record-keeping engine 335 is coupled between the logical engine 333 and the event log 245.

The security program 255 operates in conjunction with the security database 240, which includes security policies 305, known Downloadables 307, known Certificates 309 and Downloadable Security Profile (DSP) data 310 corresponding to the known Downloadables 307. Security policies 305 includes policies specific to particular users 260 and default (or generic) policies for determining whether to allow or block an incoming Downloadable. These security policies 305 may identify specific Downloadables to block, specific Downloadables to allow, or necessary criteria for allowing an unknown Downloadable. Referring to FIG. 4, security policies 305 include policy selectors 405, access control lists 410, trusted certificate lists 415, URL rule bases 420, and lists 425 of Downloadables to allow or to block per administrative override.

Known Downloadables 307 include lists of Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, of Downloadables which OEMs know to be non-hostile, and of Downloadables previously received by this security program 255. DSP data 310 includes the list of all potentially hostile or suspicious computer operations that may be attempted by each known Downloadable 307, and may also include the respective arguments of these operations. An identified argument of an operation is referred to as “resolved.” An unidentified argument is referred to as “unresolved.” DSP data 310 is described below with reference to the code scanner 325.

The ID generator 315 receives a Downloadable (including the URL from which it came and the userID of the intended recipient) from the external computer network 105 via the external communications interface 210, and generates a Downloadable ID for identifying each Downloadable. The Downloadable ID preferably includes a digital hash of the complete Downloadable code. The ID generator 315 preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the ID generator 315 may prefetch all classes embodied in or identified by the Java™ applet bytecode to generate the Downloadable ID. Similarly, the ID generator 315 may retrieve all components listed in the INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the ID generator 315 receives the same Downloadable. The ID generator 315 adds the generated Downloadable ID to the list of known Downloadables

US 6,804,780 B1

5

**307** (if it is not already listed). The ID generator **315** then forwards the Downloadable and Downloadable ID to the policy finder **317**.

The policy finder **317** uses the userID of the intended user and the Downloadable ID to select the specific security policy **305** that shall be applied on the received Downloadable. If there is a specific policy **305** that was defined for the user (or for one of its super groups) and the Downloadable, then the policy is selected. Otherwise the generic policy **305** that was defined for the user (or for one of its super groups) is selected. The policy finder **317** then sends the policy to the first comparator **320**.

The first comparator **320** receives the Downloadable, the Downloadable ID and the security policy **305** from the policy finder **317**. The first comparator **320** examines the security policy **305** to determine which steps are needed for allowing the Downloadable. For example, the security policy **305** may indicate that, in order to allow this Downloadable, it must pass all four paths, Path **1**, Path **2**, Path **3** and Path **4**. Alternatively, the security policy **305** may indicate that to allow the Downloadable, the it must pass only one of the paths. The first comparator **320** responds by forwarding the proper information to the paths identified by the security policy **305**.

#### Path 1

In path **1**, the first comparator **320** checks the policy selector **405** of the security policy **305** that was received from the policy finder **317**. If the policy selector **405** is either "Allowed" or "Blocked," then the first comparator **320** forwards this result directly to the logical engine **333**. Otherwise, the first comparator **320** invokes the comparisons in path **2** and/or path **3** and/or path **4** based on the contents of policy selector **405**. It will be appreciated that the first comparator **320** itself compares the Downloadable ID against the lists of Downloadables to allow or block per administrative override **425**. That is, the system security administrator can define specific Downloadables as "Allowed" or "Blocked."

Alternatively, the logical engine **333** may receive the results of each of the paths and based on the policy selector **405** may institute the final determination whether to allow or block the Downloadable. The first comparator **320** informs the logical engine **333** of the results of its comparison.

#### Path 2

In path **2**, the first comparator **320** delivers the Downloadable, the Downloadable ID and the security policy **305** to the code scanner **325**. If the DSP data **310** of the received Downloadable is known, the code scanner **325** retrieves and forwards the information to the ACL comparator **330**. Otherwise, the code scanner **325** resolves the DSP data **310**. That is, the code scanner **325** uses conventional parsing techniques to decompose the code (including all prefetched components) of the Downloadable into the DSP data **310**. DSP data **310** includes the list of all potentially hostile or suspicious computer operations that may be attempted by a specific Downloadable **307**, and may also include the respective arguments of these operations. For example, DSP data **310** may include a READ from a specific file, a SEND to an unresolved host, etc. The code scanner **325** may generate the DSP data **310** as a list of all operations in the Downloadable code which could ever be deemed potentially hostile and a list of all files to be accessed by the Downloadable code. It will be appreciated that the code scanner **325** may search the code for any pattern, which is undesirable or suggests that the code was written by a hacker.

6

#### An Example List of Operations Deemed Potentially Hostile

File operations: READ a file, WRITE a file;  
 Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;  
 Registry operations: READ a registry item, WRITE a registry item;  
 Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and  
 Resource usage thresholds: memory, CPU, graphics, etc.  
 In the preferred embodiment, the code scanner **325** performs a full-content inspection. However, for improved speed but reduced security, the code scanner **325** may examine only a portion of the Downloadable such as the Downloadable header. The code scanner **325** then stores the DSP data into DSP data **310** (corresponding to its Downloadable ID), and sends the Downloadable, the DSP data to the ACL comparator **330** for comparison with the security policy **305**.

The ACL comparator **330** receives the Downloadable, the corresponding DSP data and the security policy **305** from the code scanner **325**, and compares the DSP data against the security policy **305**. That is, the ACL comparator **330** compares the DSP data of the received Downloadable against the access control lists **410** in the received security policy **305**. The access control list **410** contains criteria indicating whether to pass or fail the Downloadable. For example, an access control list may indicate that the Downloadable fails if the DSP data includes a WRITE command to a system file. The ACL comparator **330** sends its results to the logical engine **333**.

#### Path 3

In path **3**, the certificate scanner **340** determines whether the received Downloadable was signed by a certificate authority, such as VeriSign, Inc., and scans for a certificate embodied in the Downloadable. The certificate scanner **340** forwards the found certificate to the certificate comparator **345**. The certificate comparator **345** retrieves known certificates **309** that were deemed trustworthy by the security administrator and compares the found certificate with the known certificates **309** to determine whether the Downloadable was signed by a trusted certificate. The certificate comparator **345** sends the results to the logical engine **333**.

#### Path 4

In path **4**, the URL comparator **350** examines the URL identifying the source of the Downloadable against URLs stored in the URL rule base **420** to determine whether the Downloadable comes from a trusted source. Based on the security policy **305**, the URL comparator **350** may deem the Downloadable suspicious if the Downloadable comes from an untrustworthy source or if the Downloadable did not come from a trusted source. For example, if the Downloadable comes from a known hacker, then the Downloadable may be deemed suspicious and presumed hostile. The URL comparator **350** sends its results to the logical engine **333**.

The logical engine **333** examines the results of each of the paths and the policy selector **405** in the security policy **305** to determine whether to allow or block the Downloadable. The policy selector **405** includes a logical expression of the results received from each of the paths. For example, the

US 6,804,780 B1

7

logical engine 333 may block a Downloadable if it fails any one of the paths, i.e., if the Downloadable is known hostile (Path 1), if the Downloadable may request suspicious operations (Path 2), if the Downloadable was not signed by a trusted certificate authority (Path 3), or if the Downloadable did come from an untrustworthy source (Path 4). The logical engine 333 may apply other logical expressions according to the policy selector 405 embodied in the security policy 305. If the policy selector 405 indicates that the Downloadable may pass, then the logical engine 333 passes the Downloadable to its intended recipient. Otherwise, if the policy selector 405 indicates that the Downloadable should be blocked, then the logical engine 333 forwards a non-hostile Downloadable to the intended recipient to inform the user that internal network security system 110 discarded the original Downloadable. Further, the logical engine 333 forwards a status report to the record-keeping engine 335, which stores the reports in event log 245 in the data storage device 230 for subsequent review, for example, by the MIS director.

FIG. 5 is a block diagram illustrating details of the security management console 120, which includes a security policy editor 505 coupled to the communications channel 135, an event log analysis engine 510 coupled between communications channel 135 and a user notification engine 515, and a Downloadable database review engine 520 coupled to the communications channel 135. The security management console 120 further includes computer components similar to the computer components illustrated in FIG. 2.

The security policy editor 505 uses an I/O interface similar to I/O interface 215 for enabling authorized user modification of the security policies 305. That is, the security policy editor 505 enables the authorized user to modify specific security policies 305 corresponding to the users 260, the default or generic security policy 305, the Downloadables to block per administrative override, the Downloadables to allow per administrative override, the trusted certificate lists 415, the policy selectors 405, the access control lists 410, the URLs in the URL rule bases 420, etc. For example, if the authorized user learns of a new hostile Downloadable, then the user can add the Downloadable to the Downloadables to block per system override.

The event log analysis engine 510 examines the status reports contained in the event log 245 stored in the data storage device 230. The event log analysis engine 510 determines whether notification of the user (e.g., the security system manager or MIS director) is warranted. For example, the event log analysis engine 510 may warrant user notification whenever ten (10) suspicious Downloadables have been discarded by internal network security system 110 within a thirty (30) minute period, thereby flagging a potential imminent security threat. Accordingly, the event log analysis engine 510 instructs the user notification engine 515 to inform the user. The user notification engine 515 may send an e-mail via internal communications interface 220 or via external communications interface 210 to the user, or may display a message on the user's display device (not shown).

FIG. 6A is a flowchart illustrating a method 600 for protecting an internal computer network 115 from suspicious Downloadables. Method 600 begins with the ID generator 315 in step 602 receiving a Downloadable. The ID generator 315 in step 604 generates a Downloadable ID identifying the received Downloadable, preferably, by generating a digital hash of the Downloadable code (including prefetched components). The policy finder 317 in step 606

8

finds the appropriate security policy 305 corresponding to the userID specifying intended recipient (or the group to which the intended recipient belongs) and the Downloadable. The selected security policy 305 may be the default security policy 305. Step 606 is described in greater detail below with reference to FIG. 6B.

The first comparator 320 in step 608 examines the lists of Downloadables to allow or to block per administrative override 425 against the Downloadable ID of the incoming Downloadable to determine whether to allow the Downloadable automatically. If so, then in step 612 the first comparator 320 sends the results to the logical engine 333. If not, then the method 600 proceeds to step 610. In step 610, the first comparator 620 examines the lists of Downloadables to block per administrative override 425 against the Downloadable ID of the incoming Downloadable for determining whether to block the Downloadable automatically. If so, then the first comparator 420 in step 612 sends the results to the logical engine 333. Otherwise, method 600 proceeds to step 614.

In step 614, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 4. If not, then method 600 jumps to step 618. If so, then the URL comparator 350 in step 616 compares the URL embodied in the incoming Downloadable against the URLs of the URL rules bases 420, and then method 600 proceeds to step 618.

In step 618, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 2. If not, then method 600 jumps to step 620. Otherwise, the code scanner 235 in step 626 examines the DSP data 310 based on the Downloadable ID of the incoming Downloadable to determine whether the Downloadable has been previously decomposed. If so, then method 600 jumps to step 630. Otherwise, the code scanner 325 in step 628 decomposes the Downloadable into DSP data. Downloadable decomposition is described in greater detail with reference to FIG. 7. In step 630, the ACL comparator 330 compares the DSP data of the incoming Downloadable against the access control lists 410 (which include the criteria necessary for the Downloadable to fail or pass the test).

In step 620, the first comparator 320 determines whether the security policy 305 indicates that the Downloadable should be tested according to Path 3. If not, then method 600 returns to step 612 to send the results of each of the test performed to the logical engine 333. Otherwise, the certificate scanner 622 in step 622 scans the Downloadable for an embodied certificate. The certificate comparator 345 in step 624 retrieves trusted certificates from the trusted certificate lists (TCL) 415 and compares the embodied certificate with the trusted certificates to determine whether the Downloadable has been signed by a trusted source. Method 600 then proceeds to step 612 by the certificate scanner 345 sending the results of each of the paths taken to the logical engine 333. The operations of the logical engine 333 are described in greater detail below with reference to FIG. 6C. Method 600 then ends.

One skilled in the art will recognize that the tests may be performed in a different order, and that each of the tests need not be performed. Further, one skilled in the art will recognize that, although path 1 is described in FIG. 6A as an automatic allowance or blocking, the results of Path 1 may be another predicate to be applied by the logical engine 333. Further, although the tests are shown serially in FIG. 6A, the tests may be performed in parallel as illustrated in FIG. 3.

US 6,804,780 B1

9

FIG. 6B is a flowchart illustrating details of step 606 of FIG. 6A (referred to herein as method 606). Method 606 begins with the policy finder 317 in step 650 determining whether security policies 305 include a specific security policy corresponding to the userID and the Downloadable. If so, then the policy finder 317 in step 654 fetches the corresponding specific policy 305. If not, then the policy finder 317 in step 652 fetches the default or generic security policy 305 corresponding to the userID. Method 606 then ends.

FIG. 6C is a flowchart illustrating details of a method 655 for determining whether to allow or to block the incoming Downloadable. Method 655 begins with the logical engine 333 in step 660 receiving the results from the first comparator 320, from the ACL comparator 330, from the certificate comparator 345 and from the URL comparator 350. The logical engine 333 in step 662 compares the results with the policy selector 405 embodied in the security policy 305, and in step 664 determines whether the policy selector 405 confirms the pass. For example, the policy selector 405 may indicate that the logical engine 333 pass the Downloadable if it passes one of the tests of Path 1, Path 2, Path 3 and Path 4. If the policy selector 405 indicates that the Downloadable should pass, then the logical engine 333 in step 666 passes the Downloadable to the intended recipient. In step 668, the logical engine 333 sends the results to the record-keeping engine 335, which in turn stores the results in the event log 245 for future review. Method 655 then ends. Otherwise, if the policy selector 405 in step 664 indicates that the Downloadable should not pass, then the logical engine 333 in step 670 stops the Downloadable and in step 672 sends a non-hostile substitute Downloadable to inform the user that the incoming Downloadable has been blocked. Method 655 then jumps to step 668.

FIG. 7 is a flowchart illustrating details of step 628 of FIG. 6A (referred to herein as method 628) for decomposing a Downloadable into DSP data 310. Method 628 begins in step 705 with the code scanner 325 disassembling the machine code of the Downloadable. The code scanner 325 in step 710 resolves a respective command in the machine code, and in step 715 determines whether the resolved command is suspicious (e.g., whether the command is one of the operations identified in the list described above with reference to FIG. 3). If not, then the code scanner 325 in step 725 determines whether it has completed decomposition of the Downloadable, i.e., whether all operations in the Downloadable code have been resolved. If so, then method 628 ends. Otherwise, method 628 returns to step 710.

Otherwise, if the code scanner 325 in step 71 determines that the resolved command is suspect, then the code scanner 325 in step 720 decodes and registers the suspicious command and its command parameters as DSP data 310. The code scanner 325 in step 720 registers the commands and command parameters into a format based on command class (e.g., file operations, network operations, registry operations, operating system operations, resource usage thresholds). Method 628 then jumps to step 725.

FIG. 8 is a flowchart illustrating a method 800 for generating a Downloadable ID for identifying a Downloadable. Method 800 begins with the ID generator 315 in step 810 receiving a Downloadable from the external computer network 105. The ID generator 315 in step 820 may fetch some or all components referenced in the Downloadable code, and in step 830 includes the fetched components in the Downloadable code. The ID generator 315 in step 840 performs a hashing function on at least a portion of the Downloadable code to generate a Downloadable ID. The ID

10

generator 315 in step 850 stores the generated Downloadable ID in the security database 240 as a reference to the DSP data 310. Accordingly, the Downloadable ID will be the same for the identical Downloadable each time it is encountered.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other variations of the above-described embodiments and methods are provided by the present invention. For example, although the invention has been described in a system for protecting an internal computer network, the invention can be embodied in a system for protecting an individual computer. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A computer-based method for generating a Downloadable ID to identify a Downloadable, comprising:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

fetching at least one software component identified by the one or more references; and

performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

2. The method of claim 1, wherein the Downloadable includes an applet.

3. The method of claim 1, wherein the Downloadable includes an active software control.

4. The method of claim 1, wherein the Downloadable includes a plugin.

5. The method of claim 1, wherein the Downloadable includes HTML code.

6. The method of claim 1, wherein the Downloadable includes an application program.

7. The method of claim 1, wherein said fetching includes fetching a first software component referenced by the Downloadable.

8. The method of claim 1, wherein said fetching includes fetching all software components referenced by the Downloadable.

9. A system for generating a Downloadable ID to identify a Downloadable, comprising:

a communications engine for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable; and

an ID generator coupled to the communications engine that fetches at least one software component identified by the one or more references, and for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

10. The system of claim 9, wherein the Downloadable includes an applet.

11. The system of claim 9, wherein the Downloadable includes an active software control.

12. The system of claim 9, wherein the Downloadable includes a plugin.

13. The system of claim 9, wherein the Downloadable includes HTML code.

US 6,804,780 B1

**11**

14. The system of claim 9, wherein the Downloadable includes an application program.

15. The system of claim 9, wherein the ID generator fetches a first software component referenced by the Downloadable.

16. The method of claim 9, wherein the ID generator fetches all software components referenced by the Downloadable.

17. A system for generating a Downloadable ID to identify a Downloadable, comprising:

means for obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

means for fetching at least one software component identified by the one or more references; and

**12**

means for performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

18. A computer-readable storage medium storing program code for causing a computer to perform the steps of:

obtaining a Downloadable that includes one or more references to software components required to be executed by the Downloadable;

5 fetching at least one software component identified by the one or more references; and

10 performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID.

\* \* \* \* \*

# **EXHIBIT 4**

**United States Patent** [19]

[11] **Patent Number:** **6,154,844**

**Touboul et al.**

[45] **Date of Patent:** **Nov. 28, 2000**

[54] **SYSTEM AND METHOD FOR ATTACHING A DOWNLOADABLE SECURITY PROFILE TO A DOWNLOADABLE**

*Primary Examiner*—Robert W. Beausoliel, Jr.  
*Assistant Examiner*—Christopher A. Revak  
*Attorney, Agent, or Firm*—Squire, Sanders & Dempsey, L.L.P.

[75] Inventors: **Shlomo Touboul**, Kefar-Haim; **Nachshon Gal**, Tel-Aviv, both of Israel

[57] **ABSTRACT**

[73] Assignee: **Finjan Software, Ltd.**, San Jose, Calif.

A system comprises an inspector and a protection engine. The inspector includes a content inspection engine that uses a set of rules to generate a Downloadable security profile corresponding to a Downloadable, e.g., Java™ applets, ActiveX™ controls, JavaScript™ scripts, or Visual Basic scripts. The content inspection engine links the Downloadable security profile to the Downloadable. The set of rules may include a list of suspicious operations, or a list of suspicious code patterns. The first content inspection engine may link to the Downloadable a certificate that identifies the content inspection engine which created the Downloadable security profile. Additional content inspection engines may generate and link additional Downloadable security profiles to the Downloadable. Each additional Downloadable security profile may also include a certificate that identifies its creating content inspection engine. Each content inspection engine preferably creates a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds. The protection includes a Downloadable interceptor for receiving a Downloadable, a file reader coupled to the interceptor for determining whether the Downloadable includes a Downloadable security profile, an engine coupled to the file reader for determining whether to trust the Downloadable security profile, and a security policy analysis engine coupled to the verification engine for comparing the Downloadable security profile against a security policy if the engine determines that the Downloadable security profile is trustworthy. A Downloadable ID verification engine retrieves the Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds, generates the Downloadable ID for the Downloadable and compares the generated Downloadable to the linked Downloadable. The protection engine further includes a certificate authenticator for authenticating the certificate that identifies a content inspection engine which created the Downloadable security profile as from a trusted source. The certificate authenticator can also authenticate a certificate that identifies a developer that created the Downloadable.

[21] Appl. No.: **08/995,648**

[22] Filed: **Dec. 22, 1997**

**Related U.S. Application Data**

[60] Provisional application No. 60/030,639, Nov. 8, 1996.

[51] **Int. Cl.**<sup>7</sup> ..... **H04L 9/36**

[52] **U.S. Cl.** ..... **713/201**; 714/38; 713/164

[58] **Field of Search** ..... 713/201, 200, 713/202, 164, 165, 166, 167, 176; 714/38, 704, 207, 33; 709/229; 380/4, 25, 24; 705/51, 54, 55

**References Cited**

**U.S. PATENT DOCUMENTS**

5,077,677	12/1991	Murphy et al.	395/10
5,359,659	10/1994	Rosenthal	380/4
5,361,359	11/1994	Tajalli et al.	395/700

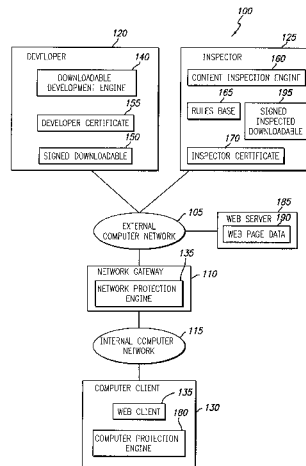
(List continued on next page.)

**OTHER PUBLICATIONS**

- X.N. Zhang, "Secure Code Distribution," Computer, pp. 76-79, Jun. 1997.
- IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, Nov. 15, 1995, pp. 6-7.
- Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, May, 1990; pp. 21-27.
- Norvin Leach et al, "IE 3.0 Applets Will Earn Certification", PC Week, v13, n29, 1998, 2 pages.
- Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996, including contents, Introduction and pp. 1-10.

(List continued on next page.)

**44 Claims, 7 Drawing Sheets**



6,154,844

Page 2

## U.S. PATENT DOCUMENTS

5,485,409	1/1996	Gupta et al.	395/186
5,485,575	1/1996	Chess et al.	395/183.14
5,572,643	11/1996	Judson	395/793
5,623,600	4/1997	Ji et al.	395/187.01
5,638,446	6/1997	Rubin	380/25
5,692,047	11/1997	McManis	380/4
5,692,124	11/1997	Holden et al.	395/187.01
5,720,033	2/1998	Deo	395/186
5,724,425	3/1998	Chang et al.	380/25
5,740,248	4/1998	Fieres et al.	380/25
5,761,421	6/1998	van Hoff et al.	395/200.53
5,765,205	6/1998	Breslau et al.	711/203
5,784,459	7/1998	Devarakonda et al.	380/4
5,796,952	8/1998	Davis et al.	395/200.54
5,805,829	9/1998	Cohen et al.	395/200.32
5,832,208	11/1998	Chen et al.	395/187.01
5,850,559	12/1998	Angelo et al.	395/750.03
5,859,966	1/1999	Hayman et al.	713/200
5,864,683	1/1999	Boebert et al.	395/200.79
5,892,904	4/1999	Atkinson et al.	713/201
5,956,481	9/1999	Walsh et al.	713/200
5,974,549	10/1999	Golan	713/200
5,983,348	11/1999	Ji	713/200

## OTHER PUBLICATIONS

Web Page, Article “Frequently Asked Questions About Authenticode”, Microsoft Corporation, last updated Feb. 17, 1997, URL: <http://www.microsoft.com/workshop/security/authcode/signfaq.asp#9>, pp. 1–13.

Web page: <http://iel.ihs.com:80/cgi-bin/iel13.cgi?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts>, Okamoto, E. et al., “ID-Based Authentication System For Computer Virus Detection”, IEEE/IEE Electronic Library online, Electronics Letters, vol. 26, Issue 15, ISSN 0013-5194, Jul. 19, 1990, Abstract and pp. 1169–1170.

“Finjan Announces a Personal Java™ Firewall for Web Browsers—the SurfinShield™ 1.6”, Press Release of Finjan Releases SurfinShield, Oct. 21, 1996, 2 pages.

“Finjan Software Releases SurfinBoard, Industry’s First JAVA Security Product For the World Wide Web”, Article published on the Internet by Finjan Software, Ltd., Jul. 29, 1996, 1 page.

“Powerful PC Security for the New World of Java™ and Downloadables, Surfin Shield™” Article published on the Internet by Finjan Software Ltd., 1996, 2 pages.

“Company Profile Finjan—Safe Surfing, The Java Security solutions Provider” Article published on the Internet by Finjan Software Ltd., Oct. 31, 1996, 3 pages.

“Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0” Las Vegas Convention Center/Pavillion 5 P5551, Nov. 18, 1996, 3 pages.

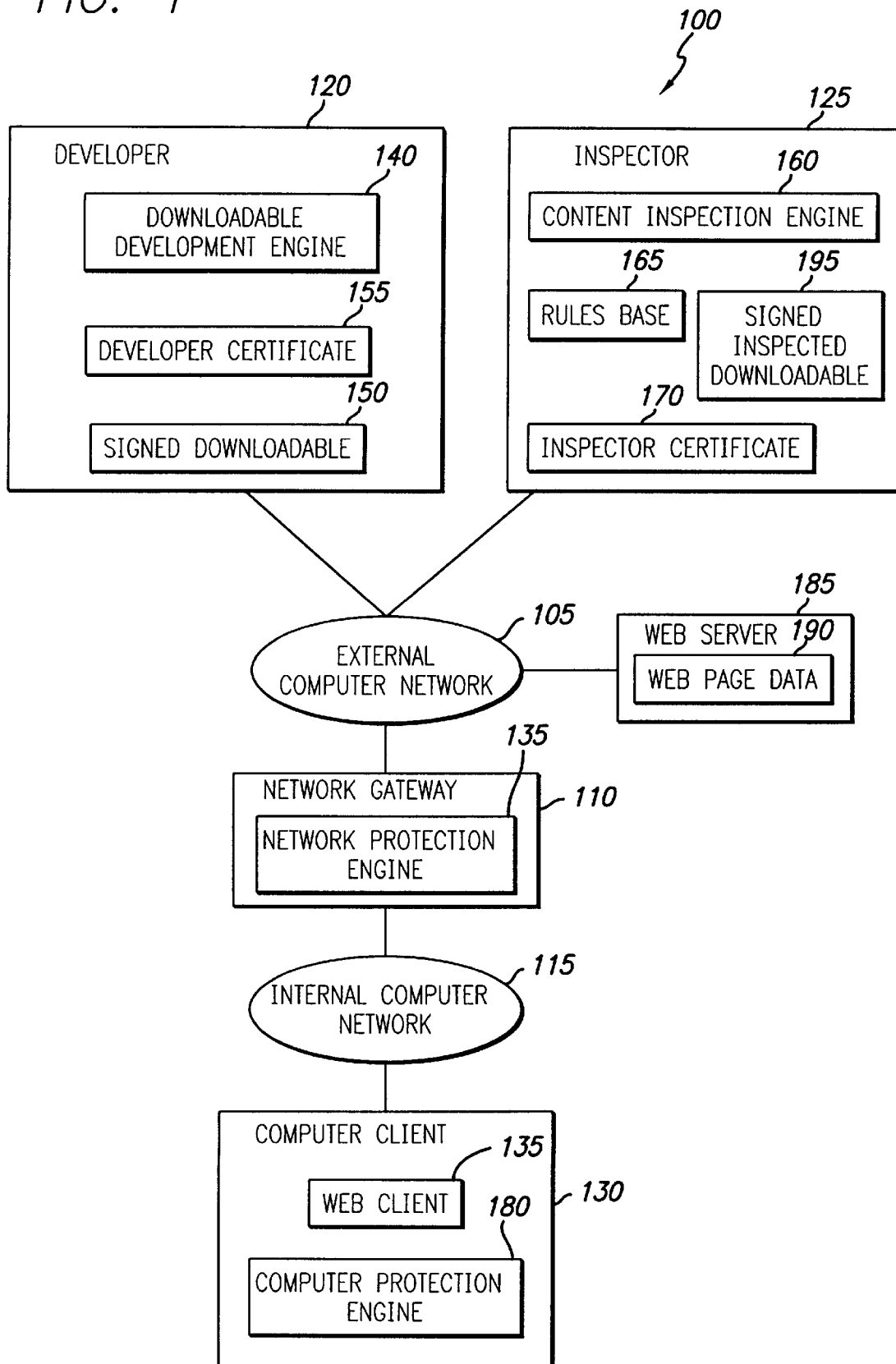
“Java Security: Issues & Solutions” Article published on the Internet by Finjan Software Ltd., 1996, 8 pages.

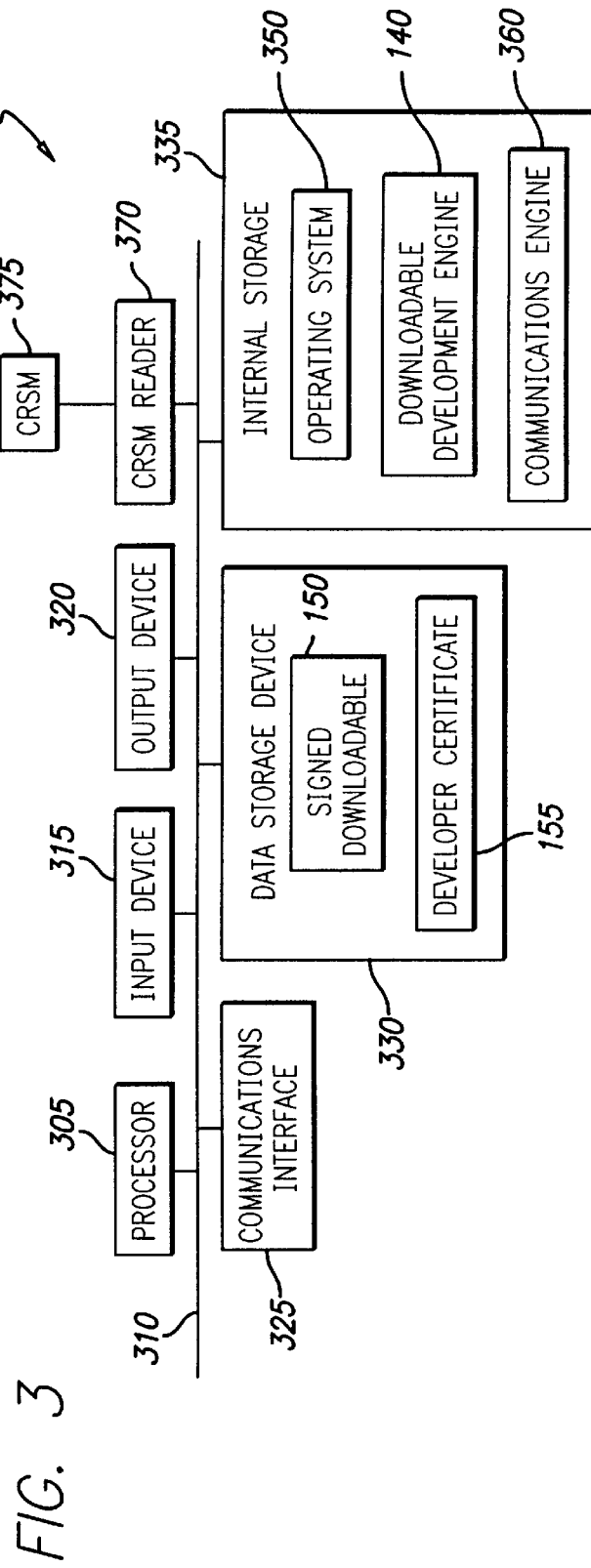
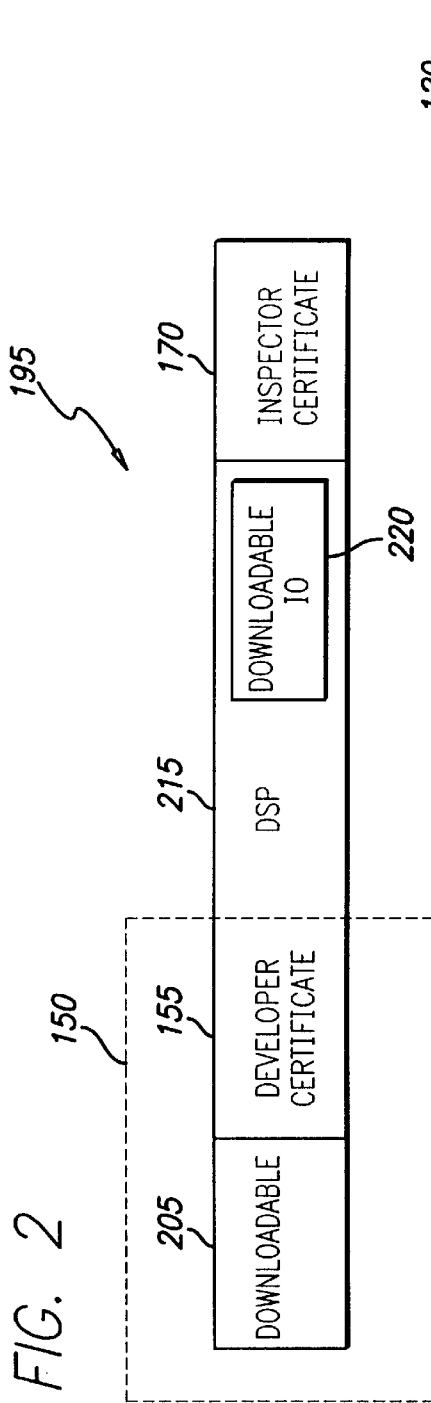
“Products” Article published on the Internet, 7 pages.

Mark LaDue, “Online Business Consultant” Article published on the Internet, Home Page, Inc. 1996, 4 pages.



FIG. 1





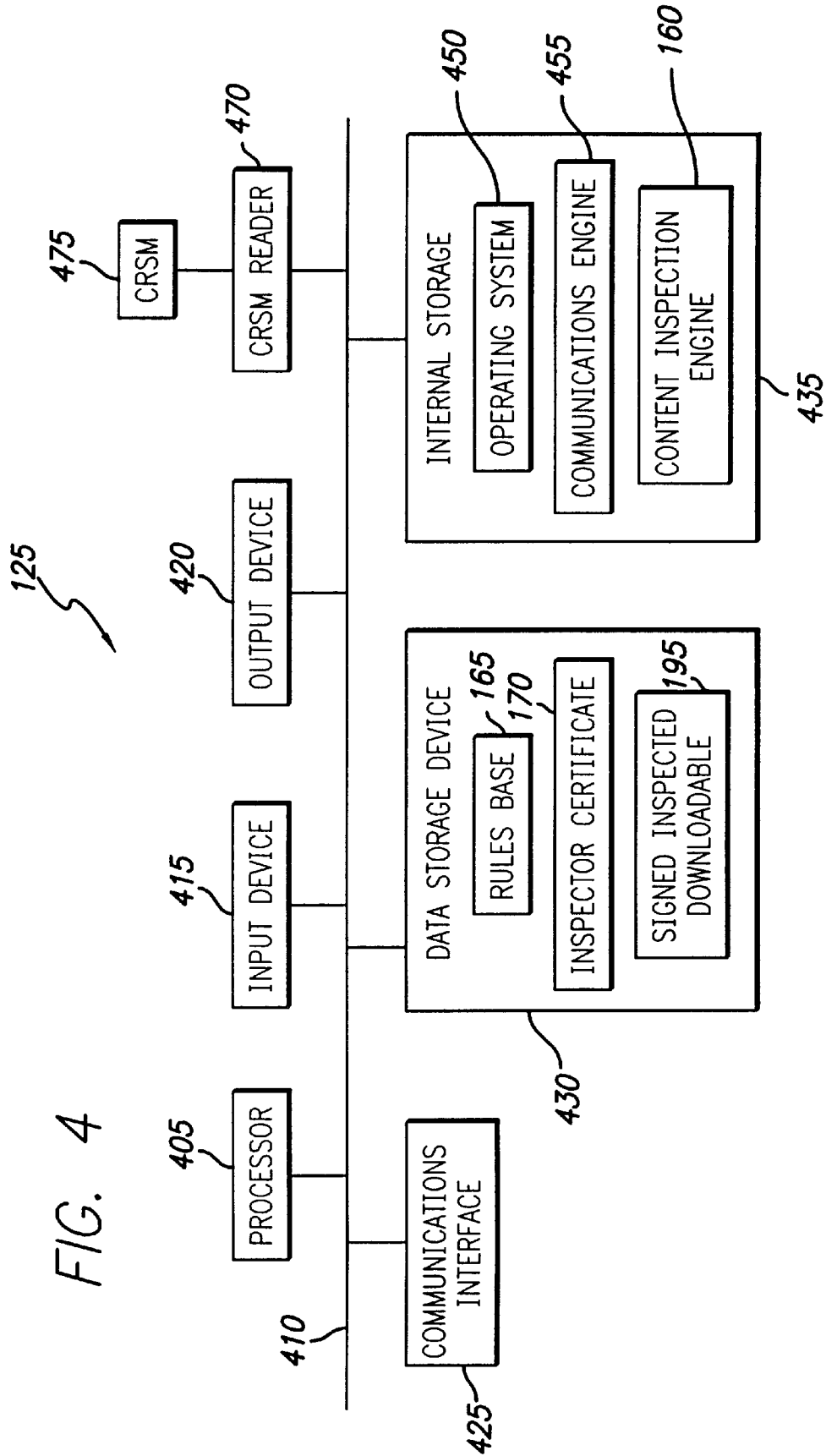


FIG. 4

FIG. 5

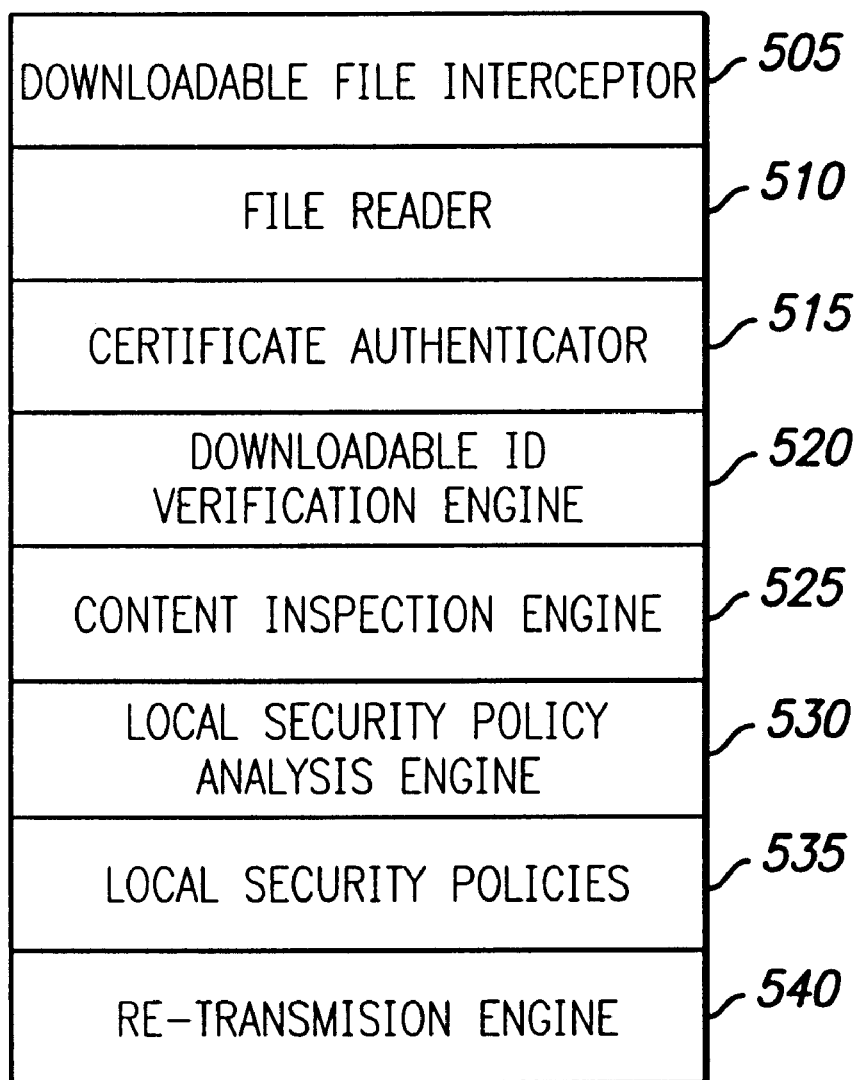


FIG. 6

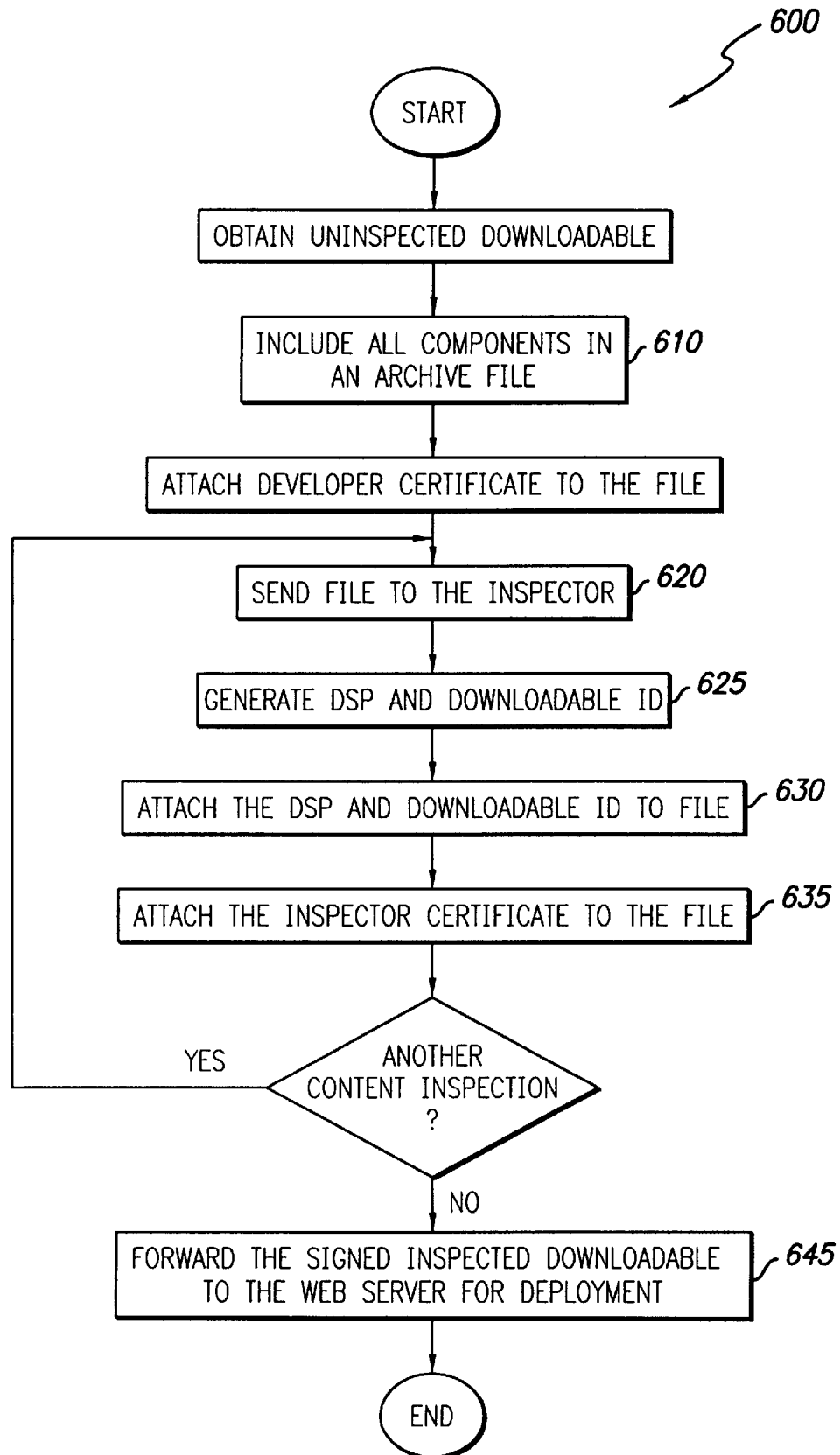
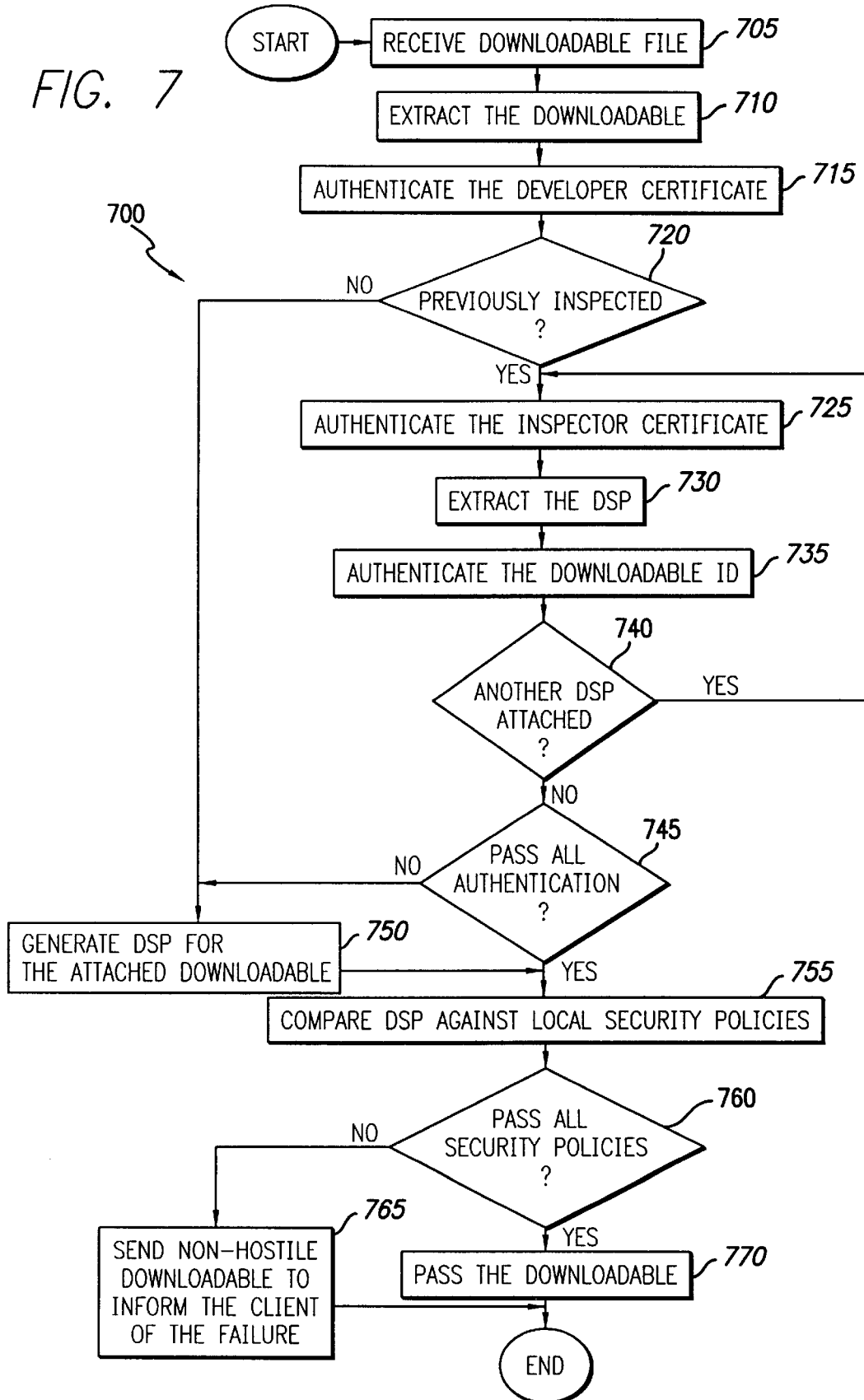


FIG. 7



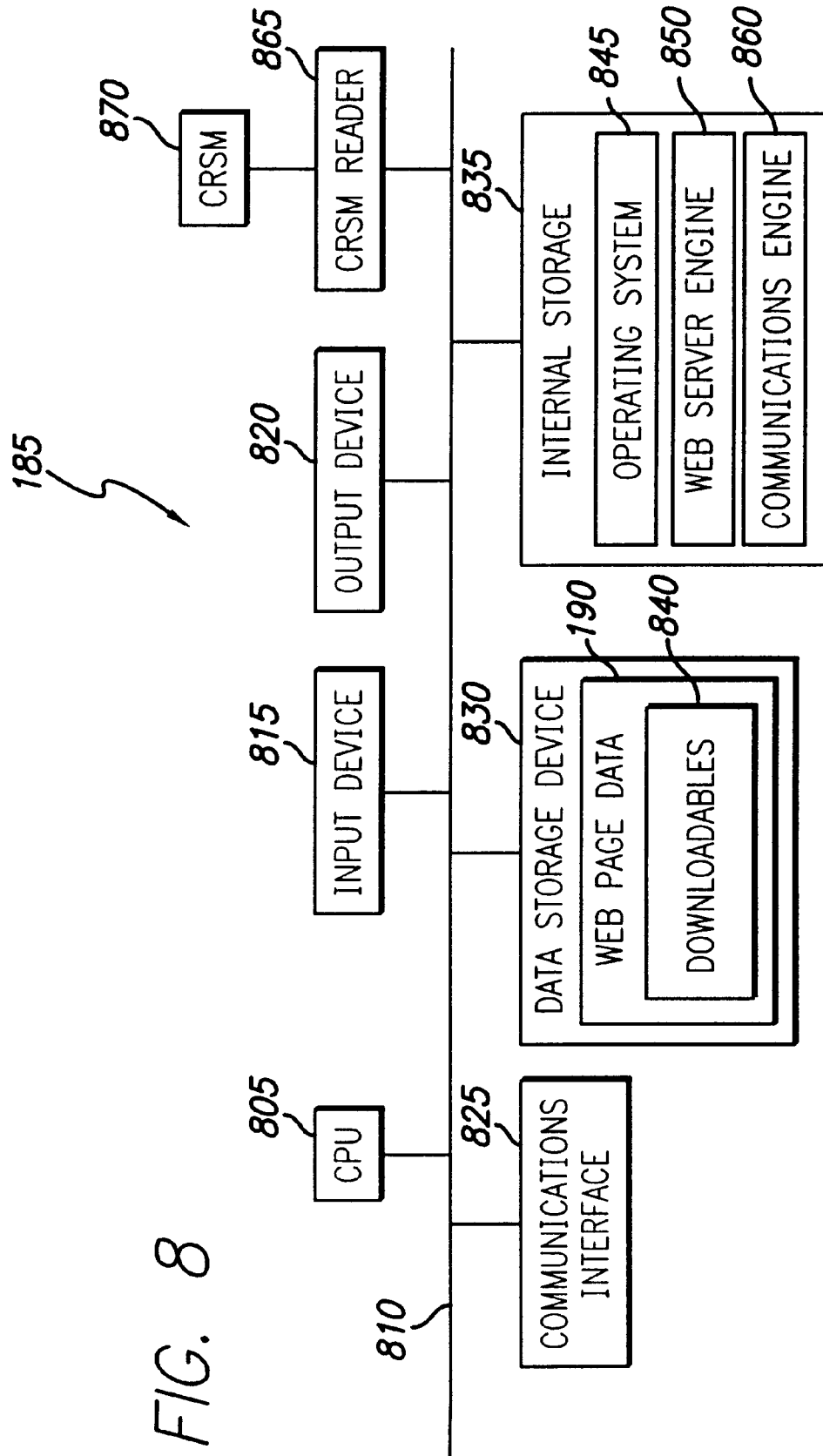


FIG. 8

6,154,844

1

## SYSTEM AND METHOD FOR ATTACHING A DOWNLOADABLE SECURITY PROFILE TO A DOWNLOADABLE

### PRIORITY REFERENCE TO RELATED APPLICATIONS

This application claims benefit of and hereby incorporates by reference provisional application Ser. No. 60/030,639, entitled "System and Method for Protecting a Computer from Hostile Downloadables," filed on Nov. 8, 1996, by inventor Shlomo Touboul; patent application Ser. No. 08/964,388, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables," filed on Nov. 6, 1997, by inventor Shlomo Touboul; and patent application Ser. No. 08/790,097, entitled "System and Method for Protecting a Client from Hostile Downloadables," filed on Jan. 29, 1997, also by inventor Shlomo Touboul.

### BACKGROUND OF THE INVENTION

#### 1. Field of the Invention

This invention relates generally to computer networks, and more particularly provides a system and method for attaching a Downloadable security profile to a Downloadable to facilitate the protection of computers and networks from a hostile Downloadable.

#### 2. Description of the Background Art

The Internet is currently a collection of over 100,000 individual computer networks owned by governments, universities, nonprofit groups and companies, and is expanding at an accelerating rate. Because the Internet is public, the Internet has become a major source of many system damaging and system fatal application programs, commonly referred to as "viruses."

Accordingly, programmers continue to design computer and computer network security systems for blocking these viruses from attacking both individual and network computers. On the most part, these security systems have been relatively successful. However, these security systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "Downloadables." A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer. A Downloadable is typically requested by an ongoing process such as by an Internet browser or web client. Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment developed by Sun Microsystems, Inc., JavaScript™ scripts also developed by Sun Microsystems, Inc., ActiveX™ controls designed for use in the ActiveX™ distributing environment developed by the Microsoft Corporation, and Visual Basic also developed by the Microsoft Corporation. Downloadables may also include plugins, which add to the functionality of an already existing application program. Therefore, a system and method are needed to protect a network from hostile Downloadables.

### SUMMARY OF THE INVENTION

The present invention provides systems for protecting a network from suspicious Downloadables, e.g., Java™ applets, ActiveX™ controls, JavaScript™ scripts, or Visual Basic scripts. The network system includes an inspector for linking Downloadable security profiles to a Downloadable, and a protection engine for examining the Downloadable

2

and Downloadable security profiles to determine whether or not to trust the Downloadable security profiles.

The inspector includes a content inspection engine that uses a set of rules to generate a Downloadable security profile corresponding to a Downloadable. The content inspection engine links the Downloadable security profile to the Downloadable. The set of rules may include a list of suspicious operations, or a list of suspicious code patterns. The first content inspection engine may link to the Downloadable a certificate that identifies the content inspection engine which created the Downloadable security profile. The system may include additional content inspection engines for generating and linking additional Downloadable security profiles to the Downloadable. Each additional Downloadable security profile may also include a certificate that identifies its creating content inspection engine. Each content inspection engine may create a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds.

The protection engine includes a Downloadable interceptor for receiving a Downloadable, a file reader coupled to the interceptor for determining whether the Downloadable includes a Downloadable security profile, an engine coupled to the file reader for determining whether to trust the Downloadable security profile, and a security policy analysis engine coupled to the verification engine for comparing the Downloadable security profile against a security policy if the engine determines that the Downloadable security profile is trustworthy. The engine preferably determines whether the first Downloadable security profile corresponds to the Downloadable. The system preferably includes a Downloadable ID verification engine for retrieving a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds. To confirm the correspondence between the Downloadable security profile and the Downloadable, the Downloadable ID verification engine generates the Downloadable ID for the Downloadable and compares the generated Downloadable to the linked Downloadable. The system may also include a content inspection engine for generating a Downloadable security profile for the Downloadable if the first Downloadable security profile is not trustworthy. The system further includes a certificate authenticator for authenticating a certificate that identifies a content inspection engine which created the Downloadable security profile as from a trusted source. The certificate authenticator can also authenticate a certificate that identifies a developer that created the Downloadable.

The present invention provides a method in a first embodiment comprising the steps of receiving a Downloadable, generating a first Downloadable security profile for the received Downloadable, and linking the first Downloadable security profile to the Downloadable. The present invention further provides a method in a second embodiment comprising the steps of receiving a Downloadable with a linked first Downloadable security profile, determining whether to trust the first Downloadable security profile, and comparing the first Downloadable security profile against the security policy if the first Downloadable security profile is trustworthy.

It will be appreciated that the system and method of the present invention may provide computer protection from known hostile Downloadables. The system and method of the present invention may identify Downloadables that perform operations deemed suspicious. The system and method of the present invention may examine the Downloadable code to determine whether the code contains any



suspicious operations, and thus may allow or block the Downloadable accordingly. It will be appreciated that, because the system and method of the present invention link a verifiable Downloadable security profile to a Downloadable, the system and method may avoid decomposing the Downloadable into the Downloadable security profile on the fly.

#### BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram illustrating a network system in accordance with the present invention;

FIG. 2 is a block diagram illustrating details of an example inspected Downloadable of FIG. 1;

FIG. 3 is a block diagram illustrating details of a developer of FIG. 1;

FIG. 4 is a block diagram illustrating details of an inspector of FIG. 1;

FIG. 5 is a block diagram illustrating details of a generic protection engine of FIG. 1;

FIG. 6 is a flowchart illustrating a method for attaching a Downloadable security profile to a Downloadable in accordance with the present invention;

FIG. 7 is a flowchart illustrating a method for examining a Downloadable in accordance with the present invention; and

FIG. 8 is a block diagram illustrating details of the web server of FIG. 1.

#### DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENT

FIG. 1 is a block diagram illustrating a computer network system **100** in accordance with the present invention. The computer network system **100** includes an external computer network **105**, such as the Wide Area Network (WAN) commonly referred to as the Internet, coupled via a network gateway **110** to an internal computer network **115**, such as a Local Area Network (LAN) commonly referred to as an intranet. The network system **100** further includes a developer **120** coupled to the external computer network **105**, an inspector **125** also coupled to the external computer network **105**, a web server **185** also coupled to the external computer network **105**, and a computer client **130** coupled to the internal computer network **115**. One skilled in the art will recognize that connections to external or internal network systems are merely exemplary, and alternative embodiments may have other connections. Further, although the developer **120**, inspector **125** and web server **185** are being described as distinct sites, one skilled in the art will recognize that these elements may be a part of an integral site, may each include components of multiple sites, or may include combinations of single and multiple sites.

The developer **120** includes a Downloadable development engine **140** for generating a signed (yet uninspected) Downloadables **150**. The developer **120** may obtain an uninspected Downloadable or may initially use the Downloadable development engine **140** to generate an uninspected Downloadable. The developer **120** can then use the Downloadable development engine **140** to transmit the signed Downloadable to the inspector **125** for hostility inspection. The developer **120** includes a developer certificate **155**, which the Downloadable development engine **140** attaches to each uninspected Downloadable so that the inspector **125**, the network gateway **110** and the computer client **130** can authenticate the developer **120**.

The inspector **125** includes a content inspection engine **160** for examining a received Downloadable, e.g., the signed

Downloadable **150** received from the developer **120**, for generating a Downloadable Security Profile (DSP) based on a rules base **165** for the Downloadable, and for attaching the DSP to the Downloadable. A DSP preferably includes a list of all potentially hostile or suspicious computer operations that may be attempted by the Downloadable, and may also include the respective arguments of these operations. Generating a DSP includes searching the Downloadable code for any pattern, which is undesirable or suggests that the code was written by a hacker. The content inspection engine **160** preferably performs a full-content inspection. It will be appreciated that generating a DSP may also include comparing a Downloadable against Downloadables which Original Equipment Manufacturers (OEMs) know to be hostile, Downloadables which OEMs know to be non-hostile, and Downloadables previously examined by the content inspection engine **160**. Accordingly, the rules base may include a list of operations and code patterns deemed suspicious, known hostile Downloadables, known viruses, etc.

#### An Example List of Operations Deemed Suspicious

File operations: READ a file, WRITE a file, DELETE a file, RENAME a file;

Network operations: LISTEN on a socket, CONNECT to a socket, SEND data, RECEIVE data, VIEW INTRANET;

Registry operations: READ a registry item, WRITE a registry item;

Operating system operations: EXIT WINDOWS, EXIT BROWSER, START PROCESS/THREAD, KILL PROCESS/THREAD, CHANGE PROCESS/THREAD PRIORITY, DYNAMICALLY LOAD A CLASS/LIBRARY, etc.; and

Resource usage thresholds: memory, CPU, graphics, etc.

Further, the content inspection engine **160** generates and attaches a Downloadable ID to the Downloadable. The Downloadable ID is typically stored as part of the DSP, since multiple DSPs may be attached to a Downloadable and each may have a different Downloadable ID. Preferably, to generate a Downloadable ID, the content inspection engine **160** computes a digital hash of the complete Downloadable code. The content inspection engine **160** preferably prefetches all components embodied in or identified by the code for Downloadable ID generation. For example, the content inspection engine **160** may prefetch all classes embodied in or identified by the Java™ applet bytecode, and then may perform a predetermined digital hash on the Downloadable code (and the retrieved components) to generate the Downloadable ID. Similarly, the content inspection engine **160** may retrieve all components listed in the .INF file for an ActiveX™ control to compute a Downloadable ID. Accordingly, the Downloadable ID for the Downloadable will be the same each time the content inspection engine **160** (or a protection engine as illustrated in FIG. 5) receives the same Downloadable and applies the same digital hash function. The downloadable components need not be stored with the Downloadable, but can be retrieved before each use or Downloadable ID generation.

Generating a DSP and generating a Downloadable ID are described in great detail with reference to the patent application Ser. No. 08/964,388, entitled "System and Method for Protecting a Computer and a Network from Hostile Downloadables," filed on Nov. 6, 1997, by inventor Shlomo Touboul, which has been incorporated by reference above.

After performing content inspection, the inspector **125** attaches an inspector certificate **170** to the Downloadable. The inspector certificate **170** verifies the authenticity of the

## 5

DSP attached to the Downloadable. Details of an example signed inspected Downloadable **150** are illustrated and described with reference to FIG. 2. The inspector **125** then transmits the signed inspected Downloadable **195** to the web server **185** for addition to web page data **190** and web page deployment. Accordingly, the computer client **130** includes a web client **175** for accessing the web page data **190** provided by the web server **185**. As is known in the art, upon recognition of a Downloadable call, the web client **175** requests the web server **185** to forward the corresponding Downloadable. The web server **185** then transmits the Downloadable via the network gateway **110** to the computer client **130**.

The network gateway **110** includes network protection engine **135**, and the computer client **130** includes a computer protection engine **180**. Both the network protection engine **135** and the computer protection engine **180** examine all incoming Downloadables and stop all Downloadables deemed suspicious. It will be appreciated that a Downloadable is deemed suspicious if it performs or may perform any undesirable operation, or if it threatens or may threaten the integrity of any computer component. It is to be understood that the term “suspicious” includes hostile, potentially hostile, undesirable, potentially undesirable, etc. Thus, if the incoming Downloadable includes a signed inspected Downloadable **195**, then the network protection engine **135** and the computer protection engine **180** can review the attached certificates to verify the authenticity of the DSP. If the incoming Downloadable does not include a signed inspected Downloadable **195**, then each of the network protection engine **135** and the computer protection engine **180** must generate the DSP, and compare the DSP against local security policies (**535**, FIG. 5).

Components and operation of the network protection engine **135** and the computer protection engine **180** are described in greater detail with reference to FIG. 5. It will be appreciated that the network gateway **110** may include the components described in the patent-application Ser. No. 08/964,388, entitled “System and Method for Protecting a Computer and a Network from Hostile Downloadables,” filed on Nov. 6, 1997, by inventor Shlomo Touboul, which has been incorporated by reference above. It will be further appreciated that the computer protection engine **180** may include the components described in the patent application Ser. No. 08/790,097, entitled “System and Method for Protecting a Client from Hostile Downloadables,” filed on Jan. 29, 1997, also by inventor Shlomo Touboul.

It will be appreciated that the network system **100** may include multiple inspectors **125**, wherein each inspector **125** may provide a different content inspection. For example, one inspector **125** may examine for suspicious operations, another inspector **125** may examine for known viruses that may be attached to the Downloadable **150**, etc. Each inspector **125** would attach a corresponding DSP and a certificate verifying the authenticity of the attached DSP. Alternatively, a single inspector **125** may include multiple content inspection engines **160**, wherein each engine provides a different content inspection.

FIG. 2 is a block diagram illustrating details of a signed inspected Downloadable **195**, which includes a Downloadable **205**, a developer certificate **155**, a DSP **215** which includes a Downloadable ID **220**, and an inspector certificate **170**. The Downloadable **205** includes the downloadable and executable code that a web client **175** receives and executes. The Downloadable **205** may be encrypted using the developer’s private key. The attached developer certificate **155** may include the developer’s public key, the devel-

## 6

oper’s name, an expiration date of the key, the name of the certifying authority that issued the certificate, and a serial number. The signed Downloadable **150** comprises the Downloadable **205** and the developer certificate **155**. The DSP **215** and Downloadable ID **220** may be encrypted by the inspector’s private key. The Downloadable ID **220** is illustrated as part of the DSP **215** for simplicity, since each signed inspected Downloadable **195** may include multiple DSPs **215** (and each DSP **215** may include a separate and distinct Downloadable ID **220**). The inspector certificate **170** may include the inspector’s public key, an expiration date of the key, the name of the certifying authority that issued the certificate, and a Ser. No.

Although the signed inspected Downloadable **195** illustrates the DSP **215** (and Downloadable ID **220**) as an attachment, one skilled in the art will recognize that the DSP **215** can be linked to the Downloadable **205** using other techniques. For example, the DSP **215** can be stored in the network system **100**, and alternatively a pointer to the DSP **215** can be attached to the signed inspected Downloadable **195**. The term “linking” herein will be used to indicate an association between the Downloadable **205** and the DSP **215** (including using a pointer from the Downloadable **195** to the DSP **215**, attaching the DSP **215** to the Downloadable **205**, etc.)

FIG. 3 is a block diagram illustrating details of the developer **120**, which includes a processor **305**, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus **310**. The developer **120** further includes an input device **315** such as a keyboard and mouse, an output device **320** such as a Cathode Ray Tube (CRT) display, a data storage device **330** such as a magnetic disk, and an internal storage **335** such as Random-Access Memory (RAM), each coupled to the signal bus **310**. A communications interface **325** couples the signal bus **325** to the external computer network **105**, as shown in FIG. 1.

An operating system **350** controls processing by processor **305**, and is typically stored in the data storage device **330** and loaded into internal storage **335** (as illustrated) for execution by processor **305**. The Downloadable development engine **140** generates signed Downloadables **150** as described above, and also may be stored in the data storage device **330** and loaded into internal storage **335** (as illustrated) for execution by processor **305**. The data storage device **330** stores the signed Downloadables **150** and the developer certificate **155**. A communications engine **360** controls communications via the communications interface **325** with the external computer network **105**, and also may be stored in the data storage device **330** and loaded into internal storage **335** (as illustrated) for execution by processor **305**.

One skilled in the art will understand that the developer **120** may also include additional information, such as network connections, additional memory, additional processors, LANs, input/output lines for transferring information across a hardware channel, the Internet or an intranet, etc. One skilled in the art will also recognize that the programs and data may be received by and stored in the system in alternative ways. For example, a computer-readable storage medium (CRSM) reader **370** such as a magnetic disk drive, hard disk drive, magneto-optical reader, CPU, etc. may be coupled to the signal bus **310** for reading a computer-readable storage medium (CRSM) **375** such as a magnetic disk, a hard disk, a magneto-optical disk, RAM, etc. Accordingly, the developer **120** may receive programs and data via the CRSM reader **370**.

FIG. 4 is a block diagram illustrating details of the inspector **125**, which includes a processor **405**, such as an

Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus **410**. The inspector **125** further includes an input device **415** such as a keyboard and mouse, an output device **420** such as a CRT display, a data storage device **430** such as a magnetic disk, and an internal storage **435** such as RAM, each coupled to the signal bus **410**. A communications interface **425** couples the signal bus **425** to the external computer network **105**, as shown in FIG. 1.

An operating system **450** controls processing by processor **405**, and is typically stored in the data storage device **430** and loaded into internal storage **435** (as illustrated) for execution by processor **405**. The content inspection engine **160** performs a content inspection of Downloadables from the developer **120** and attaches the results of the content inspection. The content inspection engine **160** also may be stored in the data storage device **330** and loaded into internal storage **335** (as illustrated) for execution by processor **405**. The data storage device **330** stores the rules base **165**, the inspector certificate **170** and the signed inspected Downloadable **195**. A communications engine **455** controls communications via the communications interface **425** with the external computer network **105**, and also may be stored in the data storage device **430** and loaded into internal storage **435** (as illustrated) for execution by processor **405**.

One skilled in the art will understand that the inspector **125** may also include additional information, such as network connections, additional memory, additional processors, LANs, input/output lines for transferring information across a hardware channel, the Internet or an intranet, etc. One skilled in the art will also recognize that the programs and data may be received by and stored in the system in alternative ways. For example, a computer-readable storage medium (CRSM) reader **470** such as a magnetic disk drive, hard disk drive, magneto-optical reader, processor, etc. may be coupled to the signal bus **410** for reading a computer-readable storage medium (CRSM) **475** such as a magnetic disk, a hard disk, a magneto-optical disk, RAM, etc. Accordingly, the inspector **125** may receive programs and data via the CRSM reader **470**.

FIG. 5 is a block diagram illustrating details of a generic protection engine **500**, which exemplifies each of the network protection engine **135** and the computer protection engine **180**. The generic protection engine **500** includes a Downloadable file interceptor **505** for intercepting incoming Downloadables (i.e., Downloadable files) for inspection, and a file reader **510** for opening the received Downloadable file and initiating appropriate components.

If the enclosed Downloadable includes a signed inspected Downloadable **195**, the file reader **510** initiates execution of a certificate authenticator **515**. The certificate authenticator **515** verifies the authenticity of the developer certificate **155** and the authenticity of the inspector certificate **170**. One skilled in the art will appreciate that certificate verification may include using authenticated or known public keys to decrypt the certificates or the enclosed files. A Downloadable ID verification engine **520** regenerates the Downloadable ID for the enclosed Downloadable **150**, and compares the regenerated Downloadable ID against the enclosed Downloadable ID **220**. If the received Downloadable fails any of the above tests (or if the received Downloadable is not a signed inspected Downloadable **195**), then a local content inspection engine **525** generates a DSP for the enclosed Downloadable **205**. Otherwise, if the received Downloadable file passes all of the above tests, then the content inspection engine **525** trusts the attached DSP **215** and thus need not generate a DSP. The content inspection

engine **525** is similar to the content inspection engine **160** of the inspector **125**. One skilled in the art will recognize that the generic protection engine **500** may include multiple content inspection engines **525** for performing distinct content examinations.

A local security policy analysis engine **530** compares the attached or generated DSP against local security policies **535**. The local security policies **535** may include a list of specific Downloadables to block, a list of specific Downloadables to allow, generic rules to apply regardless of the intended recipient and recipient's status, specific rules to apply based on the intended recipient or the intended recipient's status, trusted certificates, etc. If the received Downloadable passes all the local security policies **535**, a retransmission engine **540** passes the Downloadable onward to the intended recipient for execution.

It will be appreciated that components of the generic protection engine **500** may include components described in the patent applications incorporated by reference above. For example, the local security policies **535** in this application may include the security policies **305** of the patent application Ser. No. 08/964,388, entitled "System and Method for Protecting a Computer or a Network from Hostile Downloadables," filed on Nov. 6, 1997, by inventor Shlomo Touboul; the content inspection engine **525** and the content inspection engine **160** each may include the code scanner **325** of the same patent application; the certificate authenticator **515** may include the certificate scanner **340** and the certificate comparator **345** of the same patent application; the local security policy analysis engine **530** may include the access control lists comparator **330** and the logical engine **333** of the same patent; the Downloadable ID verification engine **520** may include the ID generator **315** of the same patent application; and the file reader **510** may include the first comparator **320** of the same patent application.

FIG. 6 is a flowchart illustrating a method **600** for inspecting a Downloadable **205** in accordance with the present invention. Method **600** begins with the developer **120** in step **605** obtaining or using the Downloadable development engine **140** to create a Downloadable **205**. The Downloadable development engine **140** in step **610** includes all components, e.g., all Java™ classes for a Java™ applet, into a Downloadable archive file, and in step **615** attaches the developer certificate **155** to the archive file, thereby creating a signed Downloadable **150**.

The Downloadable development engine **140** in step **620** transmits the signed Downloadable **150** to the inspector **125**. The content inspection engine **160** in step **625** generates a DSP **215** and a Downloadable ID **220** for the Downloadable **150**. As stated above, generating a DSP includes examining the Downloadable **205** (and the Downloadable components) for all suspicious operations that will or may be performed by the Downloadable, all suspicious code patterns, all known viruses, etc. Generating a DSP may include comparing all operations that will or may be performed against a list of suspicious operations or against a list of rules, e.g., a rules base **165**. Accordingly, if an operation in the Downloadable **205** matches one of the suspicious operations or violates one of the rules, then the operation is listed in the DSP **215**. Generating a Downloadable ID **220** includes computing a digital hash of the Downloadable **205** (and the Downloadable components), so that the Downloadable ID is identical for each instance of the same Downloadable **205**.

The content inspection engine **160** in step **630** attaches the DSP **215** and the Downloadable ID **220** to the Downloadable archive file, i.e., to the signed Downloadable **150**. The

content inspection engine 160 in step 635 attaches the inspector certificate 170 to the file, thereby providing authentication of the attached DSP 215 (including the Downloadable ID 220).

The inspector 125 in step 640 determines whether another content inspection is to be effected. If so, then method 600 returns to step 600 to send the file to the next inspector 620. One skilled in the art will recognize that the same inspector 125 may be used to perform another content inspection, attachment of another DSP 215 and Downloadable ID 220 and attachment of another inspector certificate 170. If the inspector 125 determines in step 640 that no more content inspection is to be effected, then the inspector 125 forwards the signed inspected Downloadable 195 to the web server 185 for addition to web page data 190 and web engine deployment. Accordingly, the web client 175 can access the web page data 190, and thus can retrieve the signed inspected Downloadable 195. Method 600 then ends.

FIG. 7 is a flowchart illustrating a method 700 for examining a Downloadable (whether or not signed and inspected). Method 700 begins with the Downloadable file interceptor 505 in step 705 receiving a Downloadable file. The file reader 510 in step 710 extracts the Downloadable 150, and in step 715 instructs the certificate authenticator 515 to authenticate the developer certificate 155 as from a trusted developer 120. Developer certificate authentication may include retrieving the public key for the developer 120 from a trusted source, and using the public key to decrypt the Downloadable 205.

The file reader 510 in step 720 determines whether an inspector 125 has previously inspected the received Downloadable. If the received Downloadable has not been inspected, then method 700 jumps to step 750. Otherwise, the file reader 510 initiates the certificate authenticator 515 in step 725 to authenticate the inspector certificate 170 as from a trusted inspector 125. Inspector certificate authentication may include retrieving the public key for the inspector 125 from a trusted source, and using the public key to decrypt the attached DSP 215 (including the Downloadable ID 220). The file reader 510 in step 730 extracts the DSP 215 (including the Downloadable ID 220), and in step 735 instructs the Downloadable ID verification engine 520 to authenticate the Downloadable ID 220. That is, the Downloadable ID verification engine 520 performs the same digital hash on the Downloadable 205 (including all components) to regenerate the Downloadable ID. If the components are not included in the file, then the Downloadable ID verification engine 520 may prefetch the components. Thus, if the regenerated Downloadable ID is the same as the attached Downloadable ID 220, then the Downloadable ID verification engine 520 verifies that the attached DSP 215 corresponds to the attached Downloadable 205.

The file reader 510 in step 740 determines whether another DSP 215 is attached to the file. If so, then method 700 returns to step 725. Otherwise, the content inspection engine 525 in step 745 determines whether the Downloadable 205 passed or failed any of the authentication tests above. If the Downloadable 205 failed any of the steps (or if as stated above the received Downloadable did not include a signed inspected Downloadable 195), then method 700 jumps to step 750.

The content inspection engine 525 in step 750 generates a DSP (or DSPs) for the received Downloadable as described above with reference to FIGS. 1–6. Otherwise, if the Downloadable 205 passed all the steps, then the content inspection engine 525 indicates that the DSP 215 (or DSPs

215) attached to the received Downloadable may be trusted. The local security policy analysis engine 530 in step 755 compares the DSP 215 (or DSPs 215), whether generated on the fly or extracted from the file, against local security policies 535. As stated above, the local security policies 535 may include a rules base 165 that identifies suspicious operations, suspicious code patterns, known viruses, etc. One skilled in the art will appreciate that the security policies 535 may depend on the type of DSP 215. For example, the local security policy analysis engine 530 may compare a first attached DSP 215 against the list of suspicious operations and suspicious code patterns, and may compare a second attached DSP 215 against known viruses.

The content inspection engine 160 in step 760 determines whether each DSP 215 passes all corresponding security policies 535. If each DSP 215 passes, then the local security policy analysis engine 530 in step 770 instructs the retransmission engine 540 to pass the Downloadable. If a DSP 215 fails, then the local security policy analysis engine 530 in step 765 stops the Downloadable and sends a substitute non-hostile Downloadable to the computer client 130 to inform the computer client 130 of the failure. Method 700 then ends.

FIG. 8 is a block diagram illustrating details of the web server 185, which includes a processor 805, such as an Intel Pentium® microprocessor or a Motorola Power PC® microprocessor, coupled to a signal bus 810. The web server 185 further includes an input device 815 such as a keyboard and mouse, an output device 820 such as a Cathode Ray Tube (CRT) display, a data storage device 830 such as a magnetic disk, and an internal storage 835 such as Random-Access Memory (RAM), each coupled to the signal bus 810. A communications interface 825 couples the signal bus 825 to the external computer network 105, as shown in FIG. 1.

An operating system 845 controls processing by processor 805, and is typically stored in the data storage device 830 and loaded into internal storage 835 (as illustrated) for execution by processor 805. A web server engine 850 controls web engine access to the web page data 190, and also may be stored in the data storage device 830 and loaded into internal storage 835 (as illustrated) for execution by processor 805. The data storage device 330 stores the web page data 190, which may include Downloadables 840 (whether or not signed and inspected). A communications engine 860 controls communications via the communications interface 825 with the external computer network 105, and also may be stored in the data storage device 830 and loaded into internal storage 835 (as illustrated) for execution by processor 805.

One skilled in the art will understand that the web server 185 may also include additional information, such as network connections, additional memory, additional processors, LANs, input/output lines for transferring information across a hardware channel, the Internet or an intranet, etc. One skilled in the art will also recognize that the programs and data may be received by and stored in the system in alternative ways. For example, a computer-readable storage medium (CRSM) reader 865 such as a magnetic disk drive, hard disk drive, magneto-optical reader, CPU, etc. may be coupled to the signal bus 310 for reading a computer-readable storage medium (CRSM) 870 such as a magnetic disk, a hard disk, a magneto-optical disk, RAM, etc. Accordingly, the web server 185 may receive programs and data via the CRSM reader 865.

The foregoing description of the preferred embodiments of the invention is by way of example only, and other

6,154,844

11

variations of the above-described embodiments and methods are provided by the present invention. Components of this invention may be implemented using a programmed general purpose digital computer, using application specific integrated circuits, or using a network of interconnected conventional components and circuits. Connections may be wired, wireless, modem, etc. The embodiments described herein have been presented for purposes of illustration and are not intended to be exhaustive or limiting. Many variations and modifications are possible in light of the foregoing teaching. The system is limited only by the following claims.

What is claimed is:

1. A method comprising:

receiving by an inspector a Downloadable;

generating by the inspector a first Downloadable security profile that identifies suspicious code in the received Downloadable; and

linking by the inspector the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients.

2. The method of claim 1, wherein the first Downloadable security profile is linked to the Downloadable via attachment.

3. The method of claim 1, wherein the first Downloadable security profile is linked to the Downloadable via a pointer.

4. The method of claim 1, further comprising the step of linking to the first Downloadable security profile a Downloadable ID that identifies the Downloadable.

5. The method of claim 1, wherein the Downloadable includes a Java™ applet.

6. The method of claim 1, wherein the Downloadable includes an ActiveX™ control.

7. The method of claim 1, wherein the Downloadable includes a JavaScript™ script.

8. The method of claim 1, wherein the Downloadable includes a Visual Basic script.

9. The method of claim 1, further comprising the step of linking to the Downloadable a certificate that identifies the developer which created the Downloadable.

10. The method of claim 1, further comprising the step of linking to the Downloadable a certificate that identifies a first content inspection engine which generated the first Downloadable security profile.

11. The method of claim 1, wherein the first Downloadable security profile includes a list of operations deemed suspicious by the inspector.

12. The method of claim 1, further comprising the steps of generating a second Downloadable security profile, and linking the second Downloadable security profile to the received Downloadable.

13. The method of claim 12, further comprising the steps of linking a first certificate that identifies a first content inspection engine which generated the first Downloadable security profile, and linking a second certificate that identifies a second content inspection engine which generated the second Downloadable security profile.

14. The method of claim 13, wherein each of the first Downloadable security profile and the second Downloadable security profile includes a Downloadable ID identifying the received Downloadable.

15. An inspector system comprising:

memory storing a first rule set; and

a first content inspection engine for using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the

12

Downloadable before a web server makes the Downloadable available to web clients.

16. The system of claim 15, wherein the first rule set includes a list of suspicious operations.

17. The system of claim 15, wherein the first rule set include a list of suspicious code patterns.

18. The system of claim 15, wherein the first content inspection engine links to the Downloadable a certificate that identifies the first content inspection engine which created the first Downloadable security profile.

19. The system of claim 15, further comprising a second content inspection engine for generating a second Downloadable security profile, and for linking the second Downloadable security profile to the Downloadable.

20. The system of claim 15, wherein the first content inspection engine links to the Downloadable a first certificate that identifies the first content inspection engine which created the first Downloadable security profile, and wherein the second content inspection engine links to the Downloadable a second certificate that identifies the second content inspection engine which created the second Downloadable security profile.

21. The system of claim 15, wherein the first content inspection engine creates a first Downloadable ID that identifies the Downloadable to which the first Downloadable security profile corresponds, and links the Downloadable ID to the Downloadable security profile.

22. A method performed by a network gateway comprising:

receiving a Downloadable with a linked Downloadable security profile that identifies suspicious code in the Downloadable, the Downloadable security profile being linked to the Downloadable before the web server make the Downloadable available to the web client; and

comparing the Downloadable security profile against a security policy.

23. A method performed by a network gateway comprising:

receiving a Downloadable with a linked first Downloadable security profile that identifies suspicious code in the Downloadable, the Downloadable security profile being linked to the Downloadable before the web server make the Downloadable available to the web client;

determining whether to trust the first Downloadable security profile; and

comparing the first Downloadable security profile against the security policy if the first Downloadable security profile is trustworthy.

24. The method of claim 23, wherein the step of determining includes determining whether the first Downloadable security profile corresponds to the Downloadable.

25. The method of claim 24, wherein a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds is linked to the Downloadable security profile, and wherein the step of determining includes retrieving the linked Downloadable ID.

26. The method of claim 25, further comprising the steps of generating the Downloadable ID for the Downloadable, and comparing the generated Downloadable to the linked Downloadable.

27. The method of claim 23, further comprising the step of generating a Downloadable security profile for the Downloadable if the first Downloadable security profile is not trustworthy.

13

28. The method of claim 23, wherein a certificate that identifies a content inspection engine which created the Downloadable security profile is linked to the Downloadable security profile, and wherein the step of determining includes retrieving the certificate.

29. The method of claim 28, wherein the step of determining further includes authenticating the certificate as from a trusted source.

30. The method of claim 23, wherein a certificate that identifies a developer that created the Downloadable is linked to the Downloadable, and wherein the step of determining includes retrieving the certificate.

31. The method of claim 30, wherein the step of determining further includes authenticating the certificate as from a trusted developer.

32. A network gateway system comprising:

a Downloadable interceptor for receiving a Downloadable;

a file reader coupled to the interceptor for determining whether the Downloadable includes a linked Downloadable security profile that identifies suspicious code in the Downloadable, wherein if the Downloadable includes a linked Downloadable security profile, the Downloadable was linked before the web server makes the Downloadable available to the web client;

an engine coupled to the file reader for determining whether to trust the Downloadable security profile; and

a security policy analysis engine coupled to a verification engine for comparing the Downloadable security profile against a security policy if the engine determines that the Downloadable security profile is trustworthy.

33. The system of claim 32, wherein the engine determines whether the first Downloadable security profile corresponds to the Downloadable.

34. The system of claim 33, wherein a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds is linked to the Downloadable security profile, and wherein the engine includes a Downloadable ID verification engine for retrieving the linked Downloadable ID.

35. The system of claim 34, wherein the Downloadable ID verification engine generates the Downloadable ID for the Downloadable and compares the generated Downloadable to the linked Downloadable.

36. The system of claim 32, further comprising a content inspection engine coupled to the engine for generating a Downloadable security profile for the Downloadable if the first Downloadable security profile is not trustworthy.

37. The system of claim 32, wherein a certificate that identifies a content inspection engine which created the Downloadable security profile is linked to the Downloadable security profile, and wherein the engine retrieves the certificate.

38. The system of claim 37, wherein the engine includes a certificate authenticator for authenticating the certificate as from a trusted source.

14

39. The system of claim 32, wherein a certificate that identifies a developer that created the Downloadable is linked to the Downloadable, and wherein the engine retrieves the certificate.

40. The system of claim 39, wherein the engine includes a certificate authenticator for authenticating the certificate as from a trusted developer.

41. A computer-readable storage medium storing program code for causing a data processing system on an inspector to perform the steps of:

receiving a Downloadable;

generating a first Downloadable security profile that identifies suspicious code in the received Downloadable; and

linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients.

42. A computer-readable storage medium storing program code for causing a data processing system on a network gateway to perform the steps of:

receiving a Downloadable with a linked first Downloadable security profile that identifies suspicious code in the Downloadable, the Downloadable security profile being linked to the Downloadable before the web server make the Downloadable available to the web client;

determining whether to trust the first Downloadable security profile; and

comparing the first Downloadable security profile against the security policy if the first Downloadable security profile is trustworthy.

43. An inspector system comprising:

means for receiving a Downloadable;

means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable; and

means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients.

44. A network gateway system comprising:

means for receiving a Downloadable with a linked first Downloadable security profile that identifies suspicious code in the Downloadable, the Downloadable security profile being linked to the Downloadable before the web server make the Downloadable available to the web client;

means for determining whether to trust the first Downloadable security profile; and

means for comparing the first Downloadable security profile against the security policy if the first Downloadable security profile is trustworthy.

UNITED STATES PATENT AND TRADEMARK OFFICE

**CERTIFICATE OF CORRECTION**

PATENT NO. : 6,154,844  
DATED : November 28, 2000  
INVENTOR(S) : Shlomo Touboul et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Title page.

Item [73], Assignee, **Finjan Software, Ltd.**, San Jose, Calif., after "**Finjan Software**," change the words "**Ltd.**, San Jose, Calif." to -- **Ltd.**, Kefar Haim, Israel --

Signed and Sealed this

First Day of October, 2002

Attest:



Attesting Officer

JAMES E. ROGAN  
Director of the United States Patent and Trademark Office

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 6,154,844  
APPLICATION NO. : 08/995648  
DATED : November 28, 2000  
INVENTOR(S) : Shlomo Touboul et al.

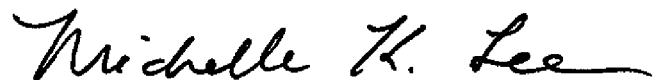
Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On The Title Page, "Related U.S. Application Data," please delete the reference to the provisional application and insert the following:

--(61) This application is a continuation-in-part of U.S. Patent Application No. 08/964,388, filed on Nov. 6, 1997, now U.S. Patent No. 6,092,194 and a continuation-in-part of U.S. Patent Application No. 08/790,097, filed on Jan. 29, 1997, now U.S. Patent No. 6,167,520.--

Signed and Sealed this  
Sixth Day of September, 2016



Michelle K. Lee  
*Director of the United States Patent and Trademark Office*





US006154844C1

(12) **EX PARTE REEXAMINATION CERTIFICATE** (10863rd)  
**United States Patent**  
**Touboul et al.** (10) **Number:** **US 6,154,844 C1**  
(45) **Certificate Issued:** **May 16, 2016**

(54) **SYSTEM AND METHOD FOR ATTACHING A DOWNLOADABLE SECURITY PROFILE TO A DOWNLOADABLE**

90/013,653, please refer to the USPTO's public Patent Application Information Retrieval (PAIR) system under the Display References tab.

(75) Inventors: **Shloma Touboul**, Kefar-Haim (IL); **Nachshon Gal**, Tel-Aviv (IL)

*Primary Examiner* — Christopher E Lee

(73) Assignee: **FINJAN, INC.**, East Palo Alto, CA (US)

(57) **ABSTRACT**

**Reexamination Request:**

No. 90/013,653, Dec. 9, 2015

A system comprises an inspector and a protection engine. The inspector includes a content inspection engine that uses a set of rules to generate a Downloadable security profile corresponding to a Downloadable, e.g., Java™ applets, ActiveX™ controls, JavaScript™ scripts, or Visual Basic scripts. The content inspection engine links the Downloadable security profile to the Downloadable. The set of rules may include a list of suspicious operations, or a list of suspicious code patterns. The first content inspection engine may link to the Downloadable a certificate that identifies the content inspection engine which created the Downloadable security profile. Additional content inspection engines may generate and link additional Downloadable security profiles to the Downloadable. Each additional Downloadable security profile may also include a certificate that identifies its creating content inspection engine. Each content inspection engine preferably creates a Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds. The protection includes a Downloadable interceptor for receiving a Downloadable, a file reader coupled to the interceptor for determining whether the Downloadable includes a Downloadable security profile, an engine coupled to the file reader for determining whether to trust the Downloadable security profile, and a security policy analysis engine coupled to the verification engine for comparing the Downloadable security profile against a security policy if the engine determines that the Downloadable security profile is trustworthy. A Downloadable ID verification engine retrieves the Downloadable ID that identifies the Downloadable to which the Downloadable security profile corresponds, generates the Downloadable ID for the Downloadable and compares the generated Downloadable to the linked Downloadable. The protection engine further includes a certificate authenticator for authenticating the certificate that identifies a content inspection engine which created the Downloadable security profile as from a trusted source. The certificate authenticator can also authenticate a certificate that identifies a developer that created the Downloadable.

**Reexamination Certificate for:**

Patent No.: **6,154,844**  
 Issued: **Nov. 28, 2000**  
 Appl. No.: **08/995,648**  
 Filed: **Dec. 22, 1997**

Certificate of Correction issued Oct. 1, 2002

**Related U.S. Application Data**

(60) Provisional application No. 60/030,639, filed on Nov. 8, 1996.

(51) **Int. Cl.**

**G06F 1/00** (2006.01)  
**G06F 21/00** (2013.01)  
**H04L 29/06** (2006.01)  
**G06F 21/51** (2013.01)  
**G06F 21/53** (2013.01)  
**G06F 21/54** (2013.01)

(52) **U.S. Cl.**

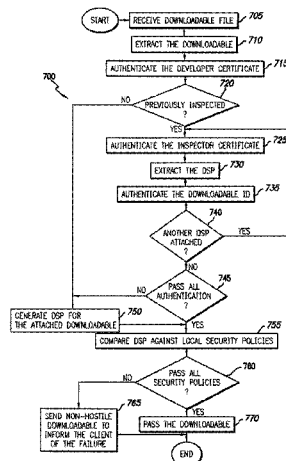
CPC ..... **G06F 21/51** (2013.01); **G06F 21/53** (2013.01); **G06F 21/54** (2013.01); **H04L 29/06** (2013.01); **H04L 63/102** (2013.01); **H04L 63/145** (2013.01); **H04L 63/168** (2013.01); **G06F 2221/009** (2013.01); **G06F 2221/033** (2013.01); **G06F 2221/2115** (2013.01); **G06F 2221/2141** (2013.01)

(58) **Field of Classification Search**

None  
 See application file for complete search history.

(56) **References Cited**

To view the complete listing of prior art documents cited during the proceeding for Reexamination Control Number



US 6,154,844 C1

1

2

**EX PARTE  
REEXAMINATION CERTIFICATE**

NO AMENDMENTS HAVE BEEN MADE TO 5  
THE PATENT

AS A RESULT OF REEXAMINATION, IT HAS BEEN  
DETERMINED THAT:

The patentability of claims **32** and **42** is confirmed. 10  
Claims **1-31**, **33-41**, **43** and **44** were not reexamined.

\* \* \* \* \*

# **EXHIBIT 5**



US008141154B2

(12) **United States Patent**  
**Gruzman et al.**

(10) **Patent No.:** **US 8,141,154 B2**  
 (45) **Date of Patent:** **Mar. 20, 2012**

(54) **SYSTEM AND METHOD FOR INSPECTING DYNAMICALLY GENERATED EXECUTABLE CODE**

(75) Inventors: **David Gruzman**, Ramat Gan (IL);  
**Yuval Ben-Itzhak**, Tel Aviv (IL)

(73) Assignee: **Finjan, Inc.** (IL)

(\* ) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 0 days.

(21) Appl. No.: **12/814,584**

(22) Filed: **Jun. 14, 2010**

(65) **Prior Publication Data**  
 US 2010/0251373 A1 Sep. 30, 2010

(51) **Int. Cl.**  
**G06F 11/00** (2006.01)  
 (52) **U.S. Cl.** ..... **726/22; 726/23; 726/24; 713/188**  
 (58) **Field of Classification Search** ..... None  
 See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,359,659	A	10/1994	Rosenthal	726/24
5,974,549	A	10/1999	Golan	726/23
5,983,348	A	11/1999	Ji	726/13
6,092,194	A	7/2000	Touboul	726/24
6,167,520	A	12/2000	Touboul	726/23
6,272,641	B1	8/2001	Ji	726/24
6,934,857	B1	8/2005	Bartleson et al.	726/5
6,965,968	B1	11/2005	Touboul	711/118
7,203,934	B2	4/2007	Souloglou et al.	717/146
7,287,279	B2	10/2007	Bertman et al.	726/23
7,313,822	B2	12/2007	Ben-Itzhak	726/24
7,739,682	B1*	6/2010	Badenell	717/174
7,836,504	B2*	11/2010	Ray et al.	726/24

2001/0005889	A1*	6/2001	Albrecht	713/201
2002/0116635	A1	8/2002	Sheymov	726/24
2004/0133796	A1	7/2004	Cohen et al.	726/24
2004/0153644	A1	8/2004	McCorkendale et al.	713/156
2004/0158729	A1	8/2004	Szor	713/190
2005/0108562	A1	5/2005	Khazan et al.	726/23
2005/0149749	A1*	7/2005	Van Brabant	713/200
2006/0015940	A1	1/2006	Zamir et al.	726/22
2006/0161981	A1	7/2006	Sheth et al.	726/22
2007/0016948	A1	1/2007	Dubrovsky et al.	726/22

\* cited by examiner

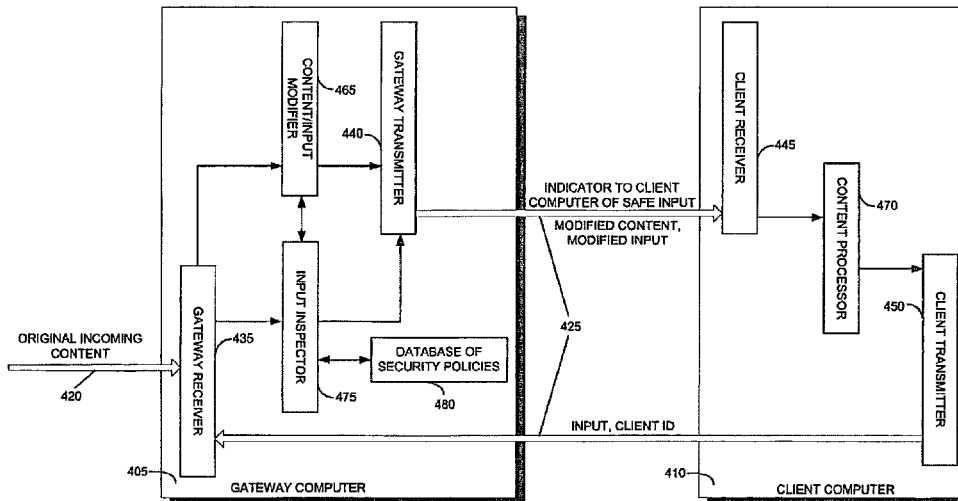
*Primary Examiner* — Ponnoreay Pich

(74) *Attorney, Agent, or Firm* — Dawn-Marie Bey; King & Spalding LLP

(57) **ABSTRACT**

A method for protecting a client computer from dynamically generated malicious content, including receiving at a gateway computer content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, modifying the content at the gateway computer, including replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input to a security computer for inspection, transmitting the modified content from the gateway computer to the client computer, processing the modified content at the client computer, transmitting the input to the security computer for inspection when the substitute function is invoked, determining at the security computer whether it is safe for the client computer to invoke the original function with the input, transmitting an indicator of whether it is safe for the client computer to invoke the original function with the input, from the security computer to the client computer, and invoking the original function at the client computer with the input, only if the indicator received from the security computer indicates that such invocation is safe. A system and a computer-readable storage medium are also described and claimed.

**12 Claims, 5 Drawing Sheets**



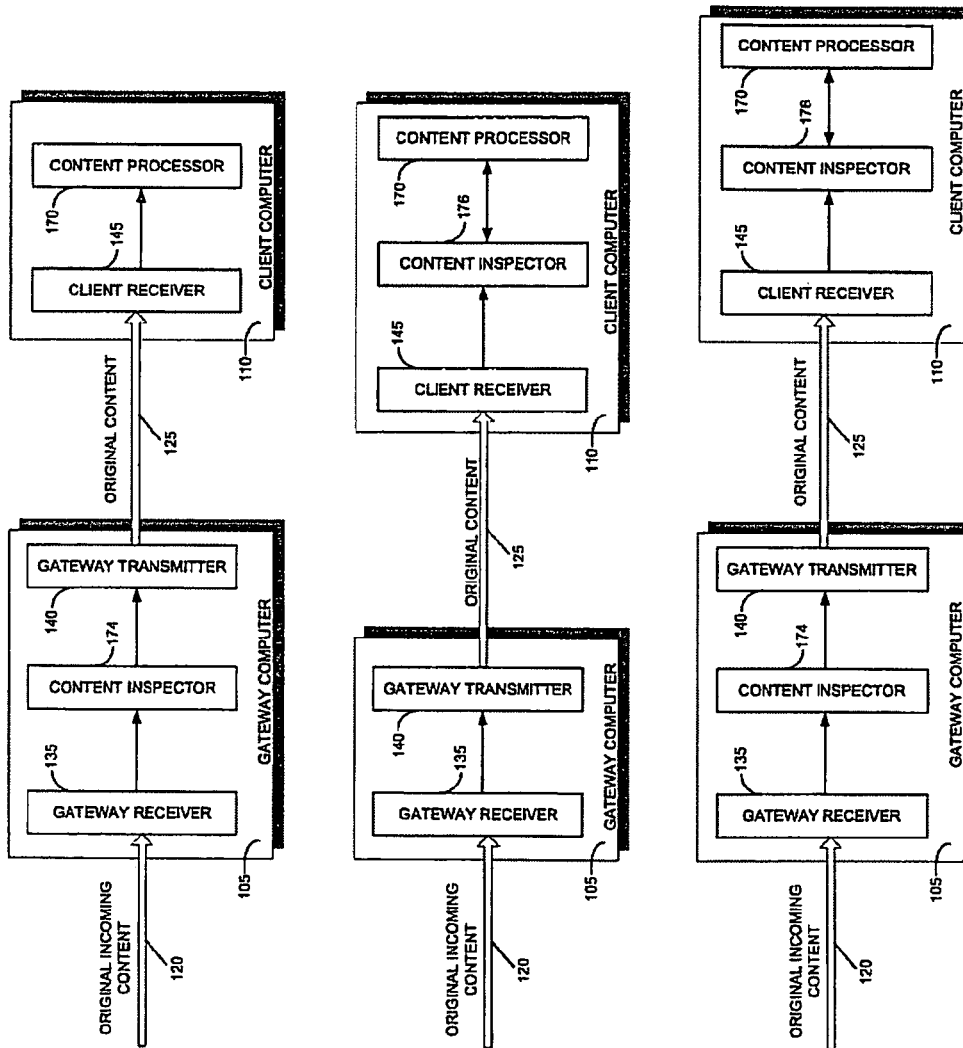


FIG. 1  
(PRIOR ART)

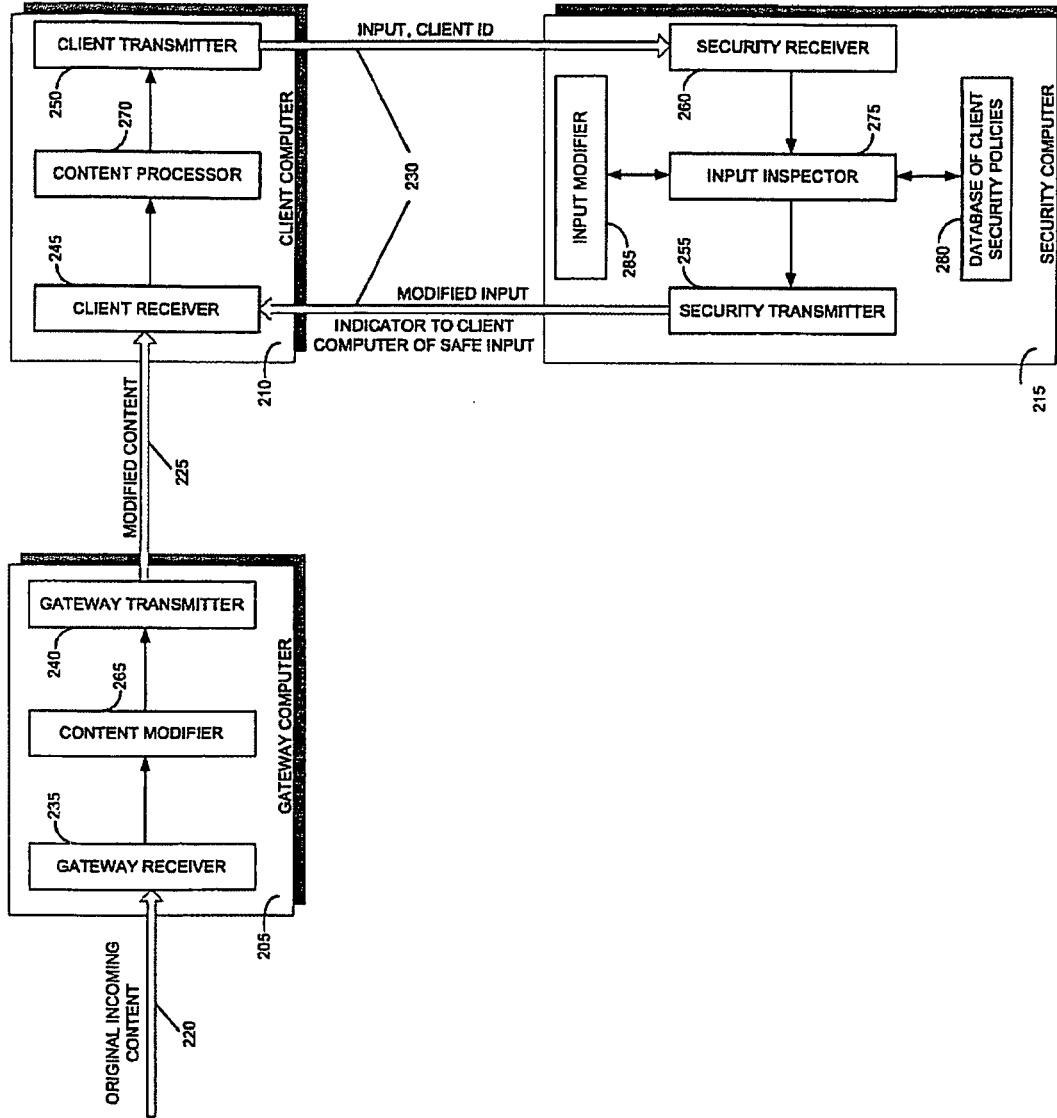


FIG. 2

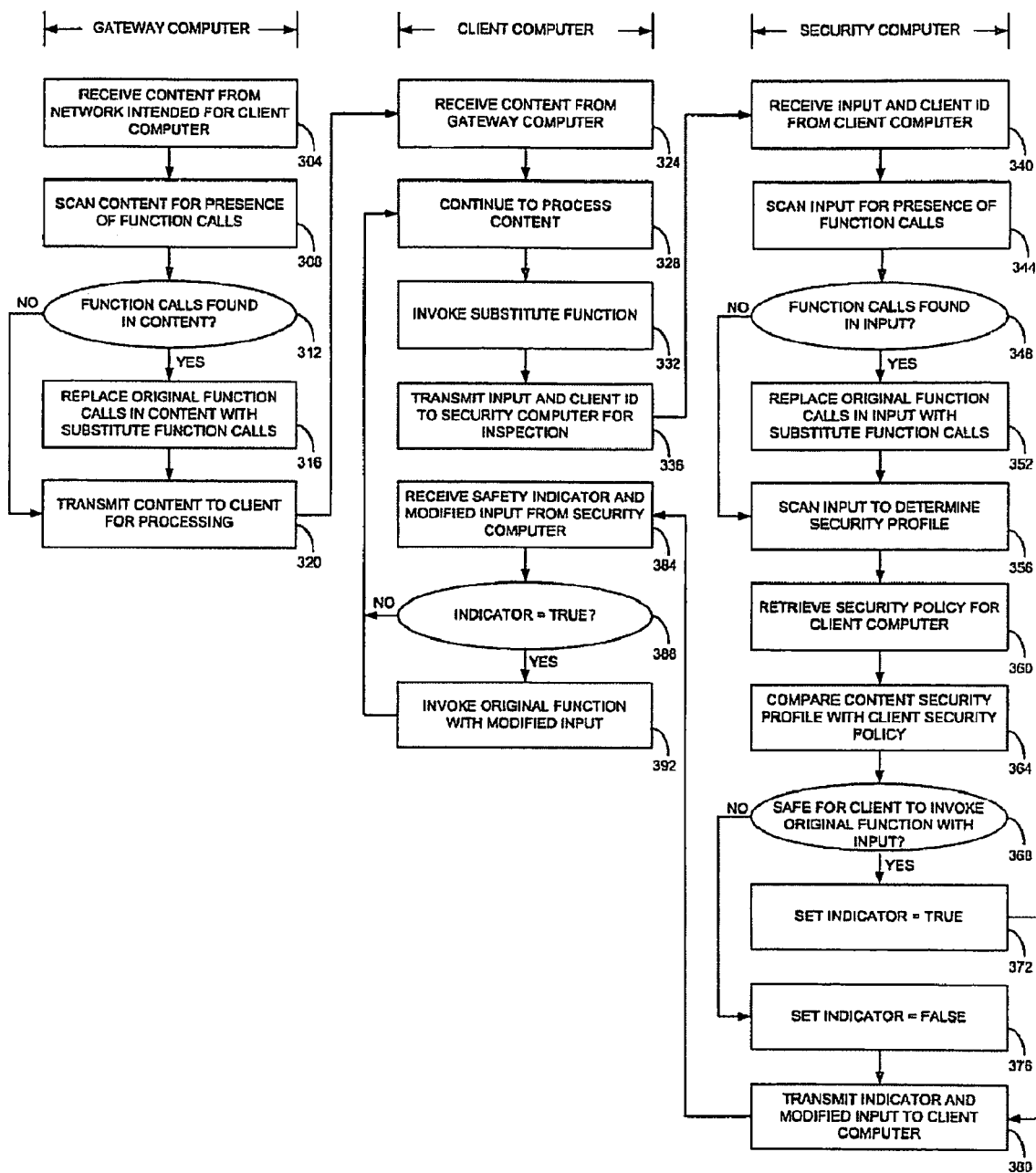


FIG. 3

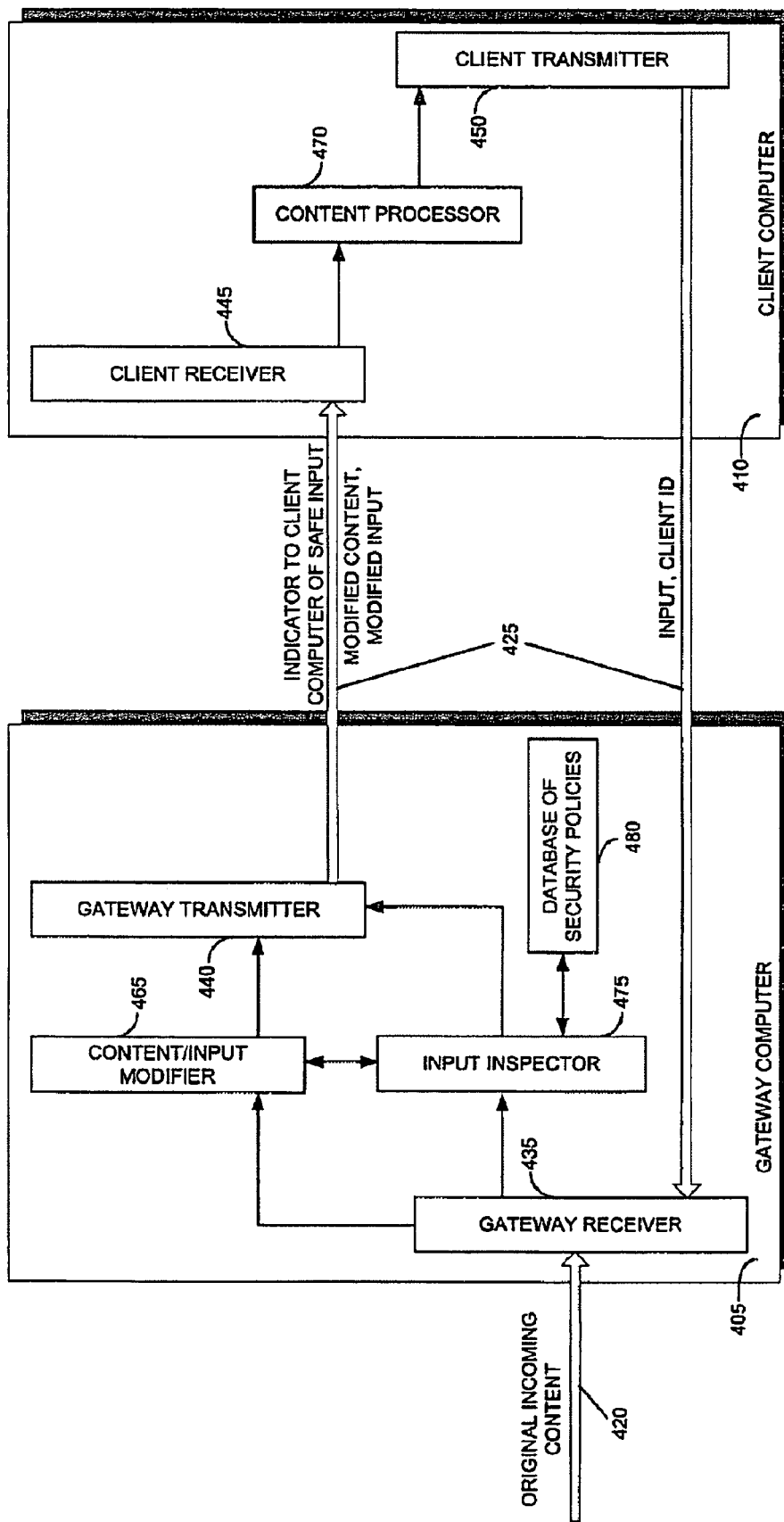


FIG. 4



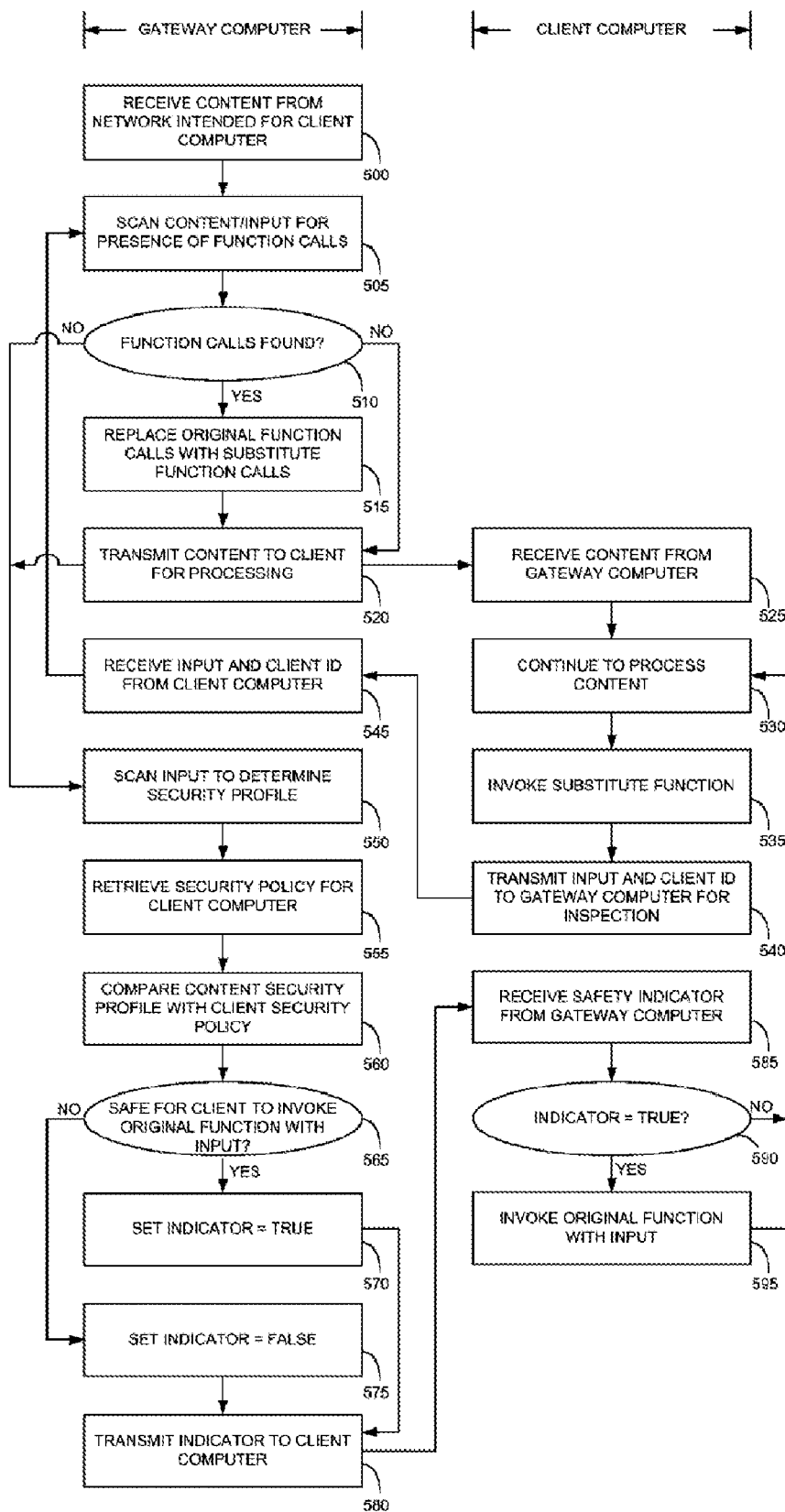


FIG. 5

US 8,141,154 B2

1

## SYSTEM AND METHOD FOR INSPECTING DYNAMICALLY GENERATED EXECUTABLE CODE

### FIELD OF THE INVENTION

The present invention relates to computer security, and more particularly to protection against malicious code such as computer viruses.

### BACKGROUND OF THE INVENTION

Computer viruses have been rampant for over two decades now. Computer viruses generally come in the form of executable code that performs adverse operations, such as modifying a computer's operating system or file system, damaging a computer's hardware or hardware interfaces, or automatically transmitting data from one computer to another. Generally, computer viruses are generated by hackers willfully, in order to exploit computer vulnerabilities. However, viruses can also arise by accident due to bugs in software applications.

Originally computer viruses were transmitted as executable code inserted into files. As each new virus was discovered, a signature of the virus was collected by anti-virus companies and used from then on to detect the virus and protect computers against it. Users began routinely scanning their file systems using anti-virus software, which regularly updated its signature database as each new virus was discovered.

Such anti-virus protection is referred to as "reactive", since it can only protect in reaction to viruses that have already been discovered.

With the advent of the Internet and the ability to run executable code such as scripts within Internet browsers, a new type of virus formed; namely, a virus that enters a computer over the Internet and not through the computer's file system. Such Internet viruses can be embedded within web pages and other web content, and begin executing within an Internet browser as soon as they enter a computer. Routine file scans are not able to detect such viruses, and as a result more sophisticated anti-virus tools had to be developed.

Two generic types of anti-virus applications that are currently available to protect against such Internet viruses are (i) gateway security applications, and (ii) desktop security applications. Gateway security applications shield web content before the content is delivered to its intended destination computer. Gateway security applications scan web content, and block the content from reaching the destination computer if the content is deemed by the security application to be potentially malicious. In distinction, desktop security applications shield against web content after the content reaches its intended destination computer.

Moreover, in addition to reactive anti-virus applications, that are based on databases of known virus signatures, recently "proactive" antivirus applications have been developed. Proactive anti-virus protection uses a methodology known as "behavioral analysis" to analyze computer content for the presence of viruses. Behavior analysis is used to automatically scan and parse executable content, in order to detect which computer operations the content may perform. As such, behavioral analysis can block viruses that have not been previously detected and which do not have a signature on record, hence the name "proactive".

Assignee's U.S. Pat. No. 6,092,194 entitled SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES, the

2

contents of which are hereby incorporated by reference, describes gateway level behavioral analysis. Such behavioral analysis scans and parses content received at a gateway and generates a security profile for the content. A security profile is a general list or delineation of suspicious, or potentially malicious, operations that executable content may perform. The derived security profile is then compared with a security policy for the computer being protected, to determine whether or not the content's security profile violates the computer's security policy. A security policy is a general set of simple or complex rules, that may be applied logically in series or in parallel, which determine whether or not a specific operation is permitted or forbidden to be performed by the content on the computer being protected. Security policies are generally configurable, and set by an administrator of the computer that is being protected.

Assignee's U.S. Pat. No. 6,167,520 entitled SYSTEM AND METHOD FOR PROTECTING A CLIENT DURING RUNTIME FROM HOSTILE DOWNLOADABLES, the contents of which are hereby incorporated by reference, describes desktop level behavioral analysis. Desktop level behavioral analysis is generally implemented during runtime, while a computer's web browser is processing web content received over the Internet. As the content is being processed, desktop security applications monitor calls made to critical systems of the computer, such as the operating system, the file system and the network system. Desktop security applications use hooks to intercept calls made to operating system functions, and allow or block the calls as appropriate, based on the computer's security policy.

Each of the various anti-virus technologies, gateway vs. desktop, reactive vs. proactive, has its pros and cons. Reactive anti-virus protection is computationally simple and fast; proactive virus protection is computationally intensive and slower. Reactive anti-virus protection cannot protect against new "first-time" viruses, and cannot protect a user if his signature file is out of date; proactive anti-virus protection can protect against new "first-time" viruses and do not require regular downloading of updated signature files. Gateway level protection keeps computer viruses at a greater distance from a local network of computers; desktop level protection is more accurate. Desktop level protection is generally available in the consumer market for hackers to obtain, and is susceptible to reverse engineering; gateway level protection is not generally available to hackers.

Reference is now made to FIG. 1, which is a simplified block diagram of prior art systems for blocking malicious content, as described hereinabove. The topmost system shown in FIG. 1 illustrates a gateway level security application. The middle system shown in FIG. 1 illustrates a desktop level security application, and the bottom system shown in FIG. 1 illustrates a combined gateway+desktop level security application.

The topmost system shown in FIG. 1 includes a gateway computer **105** that receives content from the Internet, the content intended for delivery to a client computer **110**. Gateway computer **105** receives the content over a communication channel **120**, and gateway computer communicates with client computer **110** over a communication channel **125**. Gateway computer **105** includes a gateway receiver **135** and a gateway transmitter **140**. Client computer **110** includes a client receiver **145**. Client computer generally also has a client transmitter, which is not shown.

Client computer **110** includes a content processor **170**, such as a conventional web browser, which processes Internet content and renders it for interactive viewing on a display monitor. Such Internet content may be in the form of execut-

US 8,141,154 B2

3

able code, JavaScript, VBScript, Java applets, ActiveX controls, which are supported by web browsers.

Gateway computer 105 includes a content inspector 174 which may be reactive or proactive, or a combination of reactive and proactive. Incoming content is analyzed by content inspector 174 before being transmitted to client computer 110. If incoming content is deemed to be malicious, then gateway computer 105 preferably prevents the content from reaching client computer 110. Alternatively, gateway computer 105 may modify the content so as to render it harmless, and subsequently transmit the modified content to client computer 110.

Content inspector 174 can be used to inspect incoming content, on its way to client computer 110 as its destination, and also to inspect outgoing content, being sent from client computer 110 as its origin.

The middle system shown in FIG. 1 includes a gateway computer 105 and a client computer 110, the client computer 110 including a content inspector 176. Content inspector 176 may be a conventional Signature-based anti-virus application, or a run-time behavioral based application that monitors run-time calls invoked by content processor 170 to operating system, file system and network system functions.

The bottom system shown in FIG. 1 includes both a content inspector 174 at gateway computer 105, and a content inspector 176 at client computer 110. Such a system can support conventional gateway level protection, desktop level protection, reactive anti-virus protection and proactive anti-virus protection.

As the hacker vs. anti-virus protection battle continues to wage, a newer type of virus has sprung forward; namely, dynamically generated viruses. These viruses are themselves generated only at run-time, thus thwarting conventional reactive analysis and conventional gateway level proactive behavioral analysis. These viruses take advantage of features of dynamic HTML generation, such as executable code or scripts that are embedded within HTML pages, to generate themselves on the fly at runtime.

For example, consider the following portion of a standard HTML page:

---

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0
Transitional//EN">
<HTML>
<SCRIPT LANGUAGE="JavaScript">
document.write("<h1>text that is generated at run-time</h1>");
</SCRIPT>
<BODY>
</BODY>
</HTML>
```

---

The text within the <SCRIPT> tags is JavaScript, and includes a call to the standard function document.write( ), which generates dynamic HTML. In the example above, the function document.write( ) is used to generate HTML header text, with a text string that is generated at run-time. If the text string generated at run-time is of the form <SCRIPT>malicious JavaScript</SCRIPT> then the document.write( ) function will insert malicious JavaScript into the HTML page that is currently being rendered by a web browser. In turn, when the web browser processes the inserted text, it will perform malicious operations to the client computer.

Such dynamically generated malicious code cannot be detected by conventional reactive content inspection and conventional gateway level behavioral analysis content inspection,

4

since the malicious JavaScript is not present in the content prior to run-time. A content inspector will only detect the presence of a call to Document.write( ) with input text that is yet unknown. If such a content inspector were to block all calls to Document.write( ) indiscriminately, then many harmless scripts will be blocked, since most of the time calls to Document.write( ) are made for dynamic display purposes only.

U.S. Pat. Nos. 5,983,348 and 6,272,641, both to Ji, describe reactive client level content inspection, that modifies downloaded executable code within a desktop level anti-virus application. However, such inspection can only protect against static malicious content, and cannot protect against dynamically generated malicious content.

Desktop level run-time behavioral analysis has a chance of shielding a client computer against dynamically generated malicious code, since such code will ultimately make a call to an operating system function. However, desktop anti-virus protection has a disadvantage of being widely available to the hacker community, which is always eager to find vulnerabilities. In addition, desktop anti-virus protection has a disadvantage of requiring installation of client software.

As such, there is a need for a new form of behavioral analysis, which can shield computers from dynamically generated malicious code without running on the computer itself that is being shielded.

#### SUMMARY OF THE DESCRIPTION

The present invention concerns systems and methods for implementing new behavioral analysis technology. The new behavioral analysis technology affords protection against dynamically generated malicious code, in addition to conventional computer viruses that are statically generated.

The present invention operates through a security computer that is preferably remote from a client computer that is being shielded while processing network content. During run-time, while processing the network content, but before the client computer invokes a function call that may potentially dynamically generate malicious code, the client computer passes the input to the function to the security computer for inspection, and suspends processing the network content pending a reply back from the security computer. Since the input to the function is being passed at run-time, it has already been dynamically generated and is thus readily inspected by a content inspector. Referring to the example above, were the input to be passed to the security computer prior to run-time, it would take the form of indeterminate text; whereas the input passed during run-time takes the determinate form <SCRIPT>malicious JavaScript</SCRIPT>, which can readily be inspected. Upon receipt of a reply from the security computer, the client computer resumes processing the network content, and knows whether to by-pass the function call invocation.

To enable the client computer to pass function inputs to the security computer and suspend processing of content pending replies from the security computer, the present invention operates by replacing original function calls with substitute function calls within the content, at a gateway computer, prior to the content being received at the client computer.

The present invention also provides protection against arbitrarily many recursive levels of dynamic generation of malicious code, whereby such code is generated via a series of successive function calls, one within the next.

By operating through the medium of a security computer, the present invention overcomes the disadvantages of desktop anti-virus applications, which are available to the hacker

US 8,141,154 B2

5

community for exploit. Security applications embodying the present invention are concealed securely within managed computers.

There is thus provided in accordance with a preferred embodiment of the present invention a method for protecting a client computer from dynamically generated malicious content, including receiving at a gateway computer content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, modifying the content at the gateway computer, including replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input to a security computer for inspection, transmitting the modified content from the gateway computer to the client computer, processing the modified content at the client computer, transmitting the input to the security computer for inspection when the substitute function is invoked, determining at the security computer whether it is safe for the client computer to invoke the original function with the input, transmitting an indicator of whether it is safe for the client computer to invoke the original function with the input, from the security computer to the client computer, and invoking the original function at the client computer with the input, only if the indicator received from the security computer indicates that such invocation is safe.

There is further provided in accordance with a preferred embodiment of the present invention a system for protecting a client computer from dynamically generated malicious content, including a gateway computer, including a gateway receiver for receiving content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, a content modifier for modifying the received content by replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input to a security computer for inspection, and a gateway transmitter for transmitting the modified content from the gateway computer to the client computer, a security computer, including a security receiver for receiving the input from the client computer, an input inspector for determining whether it is safe for the client computer to invoke the original function with the input, and a security transmitter for transmitting an indicator of the determining to the client computer, and a client computer communicating with the gateway computer and with the security computer, including a client receiver for receiving the modified content from the gateway computer, and for receiving the indicator from the security computer, a content processor for processing the modified content, and for invoking the original function only if the indicator indicates that such invocation is safe; and a client transmitter for transmitting the input to the security computer for inspection, when the substitute function is invoked.

There is yet further provided in accordance with a preferred embodiment of the present invention a computer-readable storage medium storing program code for causing at least one computing device to receive content including a call to an original function, and the call including an input, replace the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input for inspection, thereby generating modified content, process the modified content, transmit the input for inspection, when the substitute function is invoked while processing the modified content, and suspend processing of the modified content, determine whether it is safe to invoke the original function with the input, transmit an indicator of whether it is safe for a computer to invoke the original function with the input, and resume processing of the modified

6

content after receiving the indicator, and invoke the original function with the input only if the indicator indicates that such invocation is safe.

There is additionally provided in accordance with a preferred embodiment of the present invention a method for protecting a client computer from dynamically generated malicious content, including receiving content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, modifying the content, including replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input to a security computer for inspection, and transmitting the modified content to the client computer for processing.

There is moreover provided in accordance with a preferred embodiment of the present invention a system for protecting a client computer from dynamically generated malicious content, including a receiver for receiving content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, a content modifier for modifying the received content by replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input to a security computer for inspection, and a transmitter for transmitting the modified content to the client computer.

There is further provided in accordance with a preferred embodiment of the present invention a computer-readable storage medium storing program code for causing a computing device to receive content including a call to an original function, and the call including an input, and replace the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input for inspection.

There is yet further provided in accordance with a preferred embodiment of the present invention a method for protecting a client computer from dynamically generated malicious content, including receiving content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, modifying the content, including replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input for inspection, transmitting the modified content to the client computer for processing, receiving the input from the client computer, determining whether it is safe for the client computer to invoke the original function with the input, and transmitting to the client computer an indicator of whether it is safe for the client computer to invoke the original function with the input.

There is additionally provided in accordance with a preferred embodiment of the present invention a system for protecting a client computer from dynamically generated malicious content, including a receiver (i) for receiving content being sent to a client computer for processing, the content including a call to an original function, and the call including an input, and (ii) for receiving the input from the client computer, a content modifier for modifying the received content by replacing the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input for inspection, an input inspector for determining whether it is safe for the client computer to invoke the original function with the input, and a transmitter (i) for transmitting the modified content to the client computer, and (ii) for transmitting an indicator of the determining to the client computer.

There is moreover provided in accordance with a preferred embodiment of the present invention a computer-readable

US 8,141,154 B2

7

storage medium storing program code for causing a computing device to receive content including a call to an original function, and the call including an input, replace the call to the original function with a corresponding call to a substitute function, the substitute function being operational to send the input for inspection, and determine whether it is safe for a computer to invoke the original function with the input.

There is further provided in accordance with a preferred embodiment of the present invention a method for protecting a computer from dynamically generated malicious content, including processing content received over a network, the content including a call to a first function, and the call including an input, transmitting the input to a security computer for inspection, when the first function is invoked, receiving from the security computer an indicator of whether it is safe to invoke a second function with the input, and invoking the second function with the input, only if the indicator indicates that such invocation is safe.

There is yet further provided in accordance with a preferred embodiment of the present invention a system for protecting a computer from dynamically generated malicious content, including a content processor (i) for processing content received over a network, the content including a call to a first function, and the call including an input, and (ii) for invoking a second function with the input, only if a security computer indicates that such invocation is safe, a transmitter for transmitting the input to the security computer for inspection, when the first function is invoked, and a receiver for receiving an indicator from the security computer whether it is safe to invoke the second function with the input.

There is additionally provided in accordance with a preferred embodiment of the present invention a computer-readable storage medium storing program code for causing a computing device to process content received over a network, the content including a call to a first function, and the call including an input, transmit the input for inspection, when the first function is invoked, and suspend processing of the content, receive an indicator of whether it is safe to invoke a second function with the input, and resume processing of the content after receiving the indicator, and invoke the second function with the input only if the indicator indicates that such invocation is safe.

There is moreover provided in accordance with a preferred embodiment of the present invention a method for protecting a client computer from dynamically generated malicious content, including receiving an input from a client computer, determining whether it is safe for the client computer to invoke a function with the input, and transmitting an indicator of the determining to the client computer.

There is further provided in accordance with a preferred embodiment of the present invention a system for protecting a client computer from dynamically generated malicious content, including a receiver for receiving an input from a client computer, an input inspector for determining whether it is safe for the client computer to invoke a function with the input, and a transmitter for transmitting an indicator of the determining to the client computer.

There is further provided in accordance with a preferred embodiment of the present invention a computer-readable storage medium storing program code for causing a computing device to receive an input from a computer, determine whether it is safe for the computer to invoke a function with the input, and transmit an indicator of the determination to the computer.

The following definitions are employed throughout the specification and claims.

8

SECURITY POLICY—a set of one or more rules that determine whether or not a requested operation is permitted. A security policy may be explicitly configurable by a computer system administrator, or may be implicitly determined by application defaults.

SECURITY PROFILE—information describing one or more suspicious operations performed by executable software.

#### BRIEF DESCRIPTION OF THE DRAWINGS

The present invention will be more fully understood and appreciated from the following detailed description, taken in conjunction with the drawings in which:

FIG. 1 is a simplified block diagram of prior art systems for blocking malicious content;

FIG. 2 is a simplified block diagram of a system for protecting a computer from dynamically generated malicious executable code, in accordance with a preferred embodiment of the present invention;

FIG. 3 is a simplified flowchart of a method for protecting a computer from dynamically generated malicious executable code, in accordance with a preferred embodiment of the present invention;

FIG. 4 is a simplified block diagram of a system for protecting a computer from dynamically generated malicious executable code, in which the gateway computer itself performs the code inspection, in accordance with a preferred embodiment of the present invention; and

FIG. 5 is a simplified flowchart of a method for protecting a computer from dynamically generated malicious executable code, whereby the gateway computer itself performs the code inspection, in accordance with a preferred embodiment of the present invention.

#### DETAILED DESCRIPTION

The present invention concerns systems and methods for protecting computers against dynamically generated malicious code.

Reference is now made to FIG. 2, which is a simplified block diagram of a system for protecting a computer from dynamically generated malicious executable code, in accordance with a preferred embodiment of the present invention. Three major components of the system are a gateway computer 205, a client computer 210, and a security computer 215. Gateway computer 205 receives content from a network, such as the Internet, over a communication channel 220. Such content may be in the form of HTML pages, XML documents, Java applets and other such web content that is generally rendered by a web browser. Client computer 210 communicates with gateway computer 205 over a communication channel 225, and communicates with security computer 215 over a communication channel 230. Gateway computer 205 receives data at gateway receiver 235, and transmits data at gateway transmitter 240. Similarly, client computer 210 receives data at client receiver 245, and transmits data at client transmitter 250; and security computer 215 receives data at security receiver 260 and transmits data at security transmitter 265.

It will be appreciated by those skilled in the art that the network topology of FIG. 2 is shown as a simple topology, for purposes of clarity of exposition. However, the present invention applies to general architectures including a plurality of client computers 210 that are serviced by one or more gateway computers 205, and by one or more security computers 215. Similarly, communication channels 220, 225 and 230

9

may each be multiple channels using standard communication protocols such as TCP/IP.

Moreover, the functionality of security computer 215 may be included within gateway computer 205. Such a topology is illustrated in FIG. 4.

The computers shown in FIG. 2 also include additional processing modules, each of which is described in detail hereinbelow. Gateway computer 205 includes a content modifier 265, client computer 210 includes a content processor 270, and security computer 215 includes an inspector 275, a database of client security policies 280, and an input modifier 285.

Content modifier 265 preferably modifies original content received by gateway computer 205, and produces modified content, which includes a layer of protection to combat dynamically generated malicious code. Specifically, content modifier 265 scans the original content and identifies function calls of the form

Function(input), (1)

Content modifier 265 further modifies selected ones of the function calls (1) to corresponding function calls

Substitute\_function(input,\*), (2)

whereby the call to Function() has been replaced with a call to Substitute junction(). It is noted that the input intended for the original function is also passed to the substitute function, along with possible additional input denoted by "\*".

It will be appreciated by those skilled in the art that content modifier 265 may modify all detected function calls, or only a portion of the detected function calls. Functions that are known to be safe, regardless of their inputs, need not be modified by content modifier 265. Similarly, functions that are not passed any inputs when invoked and are known to be safe, also need not be modified by content modifier 265.

Preferably, when call (2) is made, the substitute function sends the input to security computer 215 for inspection. Preferably, content modifier 265 also inserts program code for the substitute function into the content, or a link to the substitute function. Such a substitute function may be of the following general form shown in TABLE I.

TABLE I

Generic substitute function	
Function Substitute_function(input)	
{	
inspection_result = Call_security_computer_to_inspect (	input, ID_of_client_computer);
if (inspection_result)	Original_function(input)
else	//do nothing
}	

Preferably, the above function call\_security\_computer\_to\_inspect() passes the input intended for the original function to security computer 215 for inspection by inspector 275. In addition, an ID of client computer 210 is also passed to security computer 215. For example, the ID may correspond to a network address of client computer 210. When security computer 215 services many such client computers 210 at once, it uses the IDs to determine where to return each of its many results.

Optionally, the substitute function may pass additional parameters to security computer 215, such as the name of the original function, or security policy information as described hereinbelow with reference to database 280.

10

The function call\_security\_computer\_to\_inspect() preferably returns an indicator, inspection\_result, of whether it is safe for client computer 210 to invoke the original function call (1). The indicator may be a Boolean variable, or a variable with more than two settings that can carry additional safety inspection information. In addition, as described hereinbelow with reference to input modifier 285, the function call\_security\_computer\_to\_inspect() may modify the input, and return to client computer 210 modified input to be used when invoking the original function call (1), instead of the original input. Use of input modifier 285 protects client computer 210 against recursively generated malicious code whereby the input itself to a first function generates a call to a second function.

For example, suppose a portion of the original content is of the form shown in TABLE II.

TABLE II

Example original content
<!DOCTYPE HTML PUBLIC "-//w3c//DTD HTML 4.0 Transitional//EN">
<HTML>
<SCRIPT LANGUAGE="JavaScript">
<!
Document.write("<h1>hello</h1>");
</SCRIPT>
<BODY>
</BODY>
</HTML>

Preferably, content modifier 265 alters the original content in TABLE II to the modified form shown in TABLE III. Specifically, content modifier 265 substitutes the call to the standard function Document.write(), with a call to the substitute function Substitute\_document.write(), and inserts the function definition for the substitute function into the content. The standard function Document.write() generally writes lines of HTML and inserts them into the HTML page currently being processed by a client web browser.

TABLE III

Example modified content
<!DOCTYPE HTML PUBLIC "-//w3c//DTD HTML 4.0 Transitional//EN">
<HTML>
<SCRIPT LANGUAGE="JavaScript">
<!
Function Substitute_document.write(text)
{
inspection_result = Call_security_computer_to_inspect(text);
if inspection_result
Document.write(text)
Else
//do nothing
}
Substitute_document.write("<h1>hello</h1>");
</SCRIPT>
<BODY>
</BODY>
</HTML>

Content processor 270 processes the modified content generated by content modifier 265. Content processor may be a web browser running on client computer 210. When content processor invokes the substitute function call (2), the input is passed to security computer 215 for inspection. Processing of the modified content is then suspended until security computer 215 returns its inspection results to client computer 210. Upon receiving the inspection results, client computer 210

US 8,141,154 B2

11

resumes processing the modified content. If inspection\_result is true, then client computer 210 invokes the original function call (1); otherwise, client computer 210 does not invoke the original function call (1).

Security computer 215 may also modify the input that is passed to it by the substitute function. In such case, client computer 210 invokes the original function with such modified input, instead of the original input, after receiving the inspection results.

Input inspector 275 analyzes the input passed to security computer 215 by client computer 210; specifically, the input passed when client computer 210 invokes the function call (2). Generally, input inspector 275 scans the input to determine the potentially malicious operations that it may perform, referred to as the input's "security profile". Such potentially malicious operations can include inter alia operating system level commands, file system level commands, network level commands, application level commands, certain URLs with hyperlinks, and applets already known to be malicious. Security profiles are described in assignee's U.S. Pat. No. 6,092,194 entitled SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES, the contents of which are hereby incorporated by reference. Security profiles encompass access control lists, trusted/un-trusted certificates, trusted/un-trusted URLs, and trusted/un-trusted content.

After determining a security profile for the input, inspector 275 preferably retrieves information about permission settings for client computer 210, referred to as client computer's "security policy". Such permission settings determine which commands are permitted to be performed by content processor 270 while processing content, and which commands are not permitted. Security policies are also described in assignee's U.S. Pat. No. 6,092,194. Security policies are flexible, and are generally set by an administrator of client computer 210. Preferably, security computer 215 has accesses to a database 280 of security profile information for a plurality of client computers. Database 280 may reside on security computer 215, or on a different computer.

By comparing the input's security profile to client computer 210's security policy, input inspector 275 determines whether it is safe for client computer 210 to make the function call (1). Security computer 215 sends back to client computer 210 an indicator, inspection\_result, of the inspector's determination. Comparison of a security profile to a security policy is also described in assignee's U.S. Pat. No. 6,092,194. Security policies may include simple or complex logical tests for making a determination of whether or not an input is safe.

For example, suppose the content is an HTML page, and the function call (1) is the following JavaScript:

```
Document.write("<h1><SCRIPT>Some
JavaScript</SCRIPT></h1>") (3)
```

Such a function call serves to instruct content processor 270 to insert the text between the <h1> header tags into the HTML pages; namely the text <SCRIPT>JavaScript</SCRIPT> which itself invokes the JavaScript between the <SCRIPT> tags. It is noted that the function call (1) uses a function Document.write() that is normally considered to be safe. Indeed, the function Document.write() does not access client computer 210's operating system or file system and does not send or receive data outside of client computer 210. Moreover, the input in the call (3) to Document.write() may itself be dynamically generated, and not available for inspection prior to processing the HTML page. That is, the call may be of the form

12

Document.write("content that is dynamically generated at run-time"),

where input to Document.write() may be in the form of a text string that itself is dynamically generated at run-time. Generally, such a function call cannot be analyzed successfully by behavioral based anti-virus software prior to run-time.

However, when input inspector 275 receives the input from client computer during run-time, after client computer has invoked the substitute call (2), the input has already been dynamically generated by content processor 270 and can thus be readily analyzed. Referring to the example above, when client computer 210 invokes the substitute call (2), it passes the input string

```
"<h1><SCRIPT>JavaScript</SCRIPT></h1>" (4)
```

to security computer 215. This string is then analyzed by input inspector 275, which recognizes the JavaScript and scans the JavaScript to determine any potentially malicious operations it includes. If potentially malicious operations are detected, and if they violate client computer 210's security policy, then inspector 275 preferably sets inspection\_result to false. Otherwise, inspector 275 preferably sets inspection\_result to true.

It may thus be appreciated by those skilled in the art that input inspector 275 is able to detect malicious code that is generated at runtime.

Malicious code may be generated within further recursive levels of function calls. For example, instead of the function call (3), which invokes a single function to dynamically generate JavaScript, two levels of function calls may be used. Consider, for example, the recursive function call

```
Document.write("<h1>Document.write
("<h1><SCRIPT>Some JavaScript</SCRIPT>
</h1>")</h1>") (5)
```

Such a function call first calls Document.write() to generate the function call (3), and then calls Document.write() again to generate the JavaScript. If the inputs to each of the Document.write() invocations in (5) are themselves dynamically generated at run-time, then one pass through input inspector may not detect the JavaScript.

To this end, input inspector 275 preferably passes inputs it receives to input modifier 285, prior to scanning the input. Input modifier preferably operates similar to content modifier 265, and replaces function calls detected in the input with corresponding substitute function calls. Referring to the example above, when client computer 210 invokes the outer call to Document.write() in (5), the input text string

```
"<h1>Document.write("<h1><SCRIPT>Some
JavaScript</SCRIPT></h1>")</h1>" (6)
```

is passed to security computer 215. Input modifier 285 detects the inner function call to Document.write() and replaces it with a corresponding substitute function call of the form (2). Input inspector 275 then inspects the modified input. At this stage, if the input to the inner call to Document.write() has not yet been dynamically generated, input inspector 275 may not detect the presence of the JavaScript, and thus may not set inspection\_result to false if the JavaScript is malicious. However, security computer 215 returns the modified input to client computer 210. As such, when content processor 270 resumes processing, it adds the modified input into the HTML page. This guarantees that when content processor 270 begins to process the modified input, it will again invoke the substitute function for Document.write(), which in turn passes the input of the inner Document.write() call of (5) to security

13

computer **215** for inspection. This time around input inspector **275** is able to detect the presence of the JavaScript, and can analyze it accordingly.

It may thus be appreciated by those skilled in the art that when input modifier **285** supplements input inspector **275**, inspector **275** has sufficient logic to be able to detect malicious code that is generated recursively at run-time.

In addition to inspecting inputs, security computer **215** preferably maintains an event log of potential security breaches. When input inspector **275** determines that an input is riot safe, security computer **215** enters information about the input and client computer **210** into a log that is available for review by an administrator of client computer **210**.

In accordance with a preferred embodiment of the present invention, it is anticipated that many client computers **210** use the same security computer **215** for protection. Each client computer may independently send inputs to security computer **215** for inspection. Security computer **215** may use cache memory to save results of inspection, so as to obviate the need to analyze the same input more than once. Use of cache memory when working with a plurality of security policies is described in assignee's U.S. Pat. No. 6,965,968 entitled POLICY-BASED CACHING.

Similarly, it is anticipated that gateway computer **205** services many client computers **210**. Gateway computer may include its own content inspector, which is useful for detecting malicious content that is not dynamically generated, as described in assignee's U.S. Pat. No. 6,092,194.

It may be appreciated that substitute functions as in TABLE I may also pass the name of the original function to the security computer. That is, the call to `Call_security_computer_to_inspect()` may also pass a variable, say `name_of_function`, so that input inspector **275** can determine whether it is safe to invoke the specific original function with the input. In this way, input inspector **275** can distinguish between different functions with the same input.

Reference is now made to FIG. 3, which is a simplified flowchart of a method for protecting a computer from dynamically generated malicious executable code, in accordance with a preferred embodiment of the present invention. The leftmost column of FIG. 3 shows steps performed by a gateway computer, such as gateway computer **205**. The middle column of FIG. 3 shows steps performed by a client computer, such as client computer **210**. The rightmost column of FIG. 3 shows steps performed by a security computer, such as security computer **215**.

At step **304**, the gateway computer receives content from a network, the content on its way for delivery to the client computer. Such content may be in the form of an HTML web page, an XML document, a Java applet, an EXE file, JavaScript, VBScript, an ActiveX Control, or any such data container that can be rendered by a client web browser. At step **308**, the gateway computer scans the content it received, for the presence of function calls. At step **312**, the gateway computer branches, depending on whether or not function calls were detected at step **308**. If function calls were detected, then at step **316** the gateway computer replaces original function calls with substitute function calls within the content, thereby modifying the content. If function calls were not detected, then the gateway computer skips step **316**. At step **320**, the gateway computer sends the content, which may have been modified at step **316**, to the client computer.

At step **324** the client computer receives the content, as modified by the gateway computer. At step **328** the client computer begins to continuously process the modified content; i.e., the client computer runs an application, such as a web browser or a Java virtual machine, that processes the

14

modified content. At step **332**, while processing the modified content, the client computer encounters a call (2) to a substitute function, such as the substitute function listed in TABLE I. Client computer then transmits the input to the substitute function and an identity of the client computer, to the security computer for inspection, at step **336**. The identity of the client computer serves to inform the security computer where to return its inspection result. Since one security computer typically services many client computers, passing client computer identities is a way to direct the security computer where to send back its results. At this point, client computer suspends processing the modified content pending receipt of the inspection results from the security computer. As mentioned hereinabove, the client computer may also send the name of the original function to the security computer, for consideration in the inspection analysis.

At step **340** the security computer receives the input and client computer identifier. At step **344** the security computer scans the input for the presence of function calls. At step **348** the security computer branches, depending on whether or not function calls were detected at step **344**. If function calls were detected, then the security computer replaces original function calls with substitute function calls at step **352**, thereby modifying the input. The security computer may insert definitions of the substitute functions into the input, as indicated in TABLE III, or may insert links to such definitions. Otherwise, the security computer skips step **352**. Steps **344**, **348** and **352** are similar to respective steps **308**, **312** and **316** performed by the gateway computer.

At step **356** the security computer scans the input, which may have been modified at step **352**, for the presence of potentially malicious operations. Preferably, the security computer determines a security profile for the input, which corresponds to a list of the potentially malicious operations that are detected.

At step **360** the security computer retrieves a security policy that governs the client computer. The security policy may be retrieved from a database that stores a plurality of security policies, each policy configurable by an administrator of client computers. Security policies may be set at a fine granularity of a policy for each client computer, or at a coarser granularity of a policy that applies to an entire department or workgroup.

At step **364** the security computer compares the security profile of the input under inspection with the security policy of the client computer, to determine if it is permissible for the client computer to invoke an original function with the input. Such determination may involve one or more simple or complex logical tests, structured in series or in parallel, or both, as described in assignee's U.S. Pat. No. 6,092,194.

At step **368** the security computer branches depending on the result of the comparison step **364**. If the comparison step determines that the input is safe; i.e., that the input's security profile does not violate the client computer's security policy, then at step **372** the security computer sets an indicator of inspection results to true. Otherwise, at step **376** the security computer sets the indicator to false. At step **380** the security computer returns the indicator to the client computer. In addition, if the security computer modified the input at step **352**, then it also returns the modified input to the client computer.

At step **384** the client computer receives the indicator and the modified input from the security computer and resumes processing the modified content, which had been suspended after step **336** as described hereinabove. At step **388** the client computer branches depending on the value of the indicator it received from the security computer. If the indicator is true, indicating that it is safe for the client computer to invoke the



15

original function call (1), then the client computer invokes the original function using the modified input it received from the security computer, at step 392. Otherwise, the client computer does not invoke the original function, since the indicator indicates that such invocation may be malicious to the client computer. The client computer then loops back to step 328 to continue processing the modified content.

As described hereinabove, steps 344, 348 and 352, which modify the input, are useful in protecting against malicious code that is dynamically generated in a recursive manner, as in function call (5). The security computer may require multiple passes to detect such malicious code, and steps 344, 348 and 352 provide the mechanism for this to happen.

Reference is now made to FIG. 4, which is a simplified block diagram of a system for protecting a computer from dynamically generated malicious executable code, in which the gateway computer itself performs the code inspection, in accordance with a preferred embodiment of the present invention. The system illustrated in FIG. 4 is similar to the system of FIG. 2, where the functionality of the security computer has been incorporated into the gateway computer. The elements in FIG. 4 are thus similar in functionality to the elements in FIG. 2.

Two major components of the system, gateway computer 405 and client computer 410 communicate back and forth over communication channel 425. Gateway computer 405 includes a gateway receiver 435 and a gateway transmitter 440; and client computer 410 includes a client receiver 445 and a client transmitter 450. Although FIG. 4 includes only one client computer, this is solely for the purpose of clarity of exposition, and it is anticipated that gateway computer 405 serves many client computers 410.

Gateway computer 405 receives content, such as web content, from a network, over communication channel 420. Client computer 410 includes a content processor 470, such as a web browser, which processes content received from the network.

In accordance with a preferred embodiment of the present invention, gateway computer 405 includes an input inspector 475, and a content modifier 465 which also serves as an input modifier. That is, content modifier 465 incorporates the functionalities of content modifier 265 and input modifier 285 from FIG. 2. In addition, gateway computer 405 includes a database 480 of security policies, or else has access to such a database. The operations of input inspector 475 and content/input modifier 465 are similar to the operations of the corresponding elements in FIG. 2, as described hereinabove.

Incoming content received at gateway computer 405 passes through content modifier 465, which replaces function calls of the form (1) with substitute function calls of the form (2), and the modified content is transmitted to client computer 410. Content processor 470 processes the modified content and, while processing the modified content, if it encounters a substitute function call it sends the function's input to inspector 475 for inspection, and suspends processing of the modified content. The input passes through input modifier 465, and input inspector 475 analyzes the modified input for the presence of potentially malicious operations. Gateway computer 405 returns the input inspection results to client computer 410. Gateway computer 405 may also return the modified input to client computer 410. After receiving the inspection results, client computer 410 resumes processing the modified content and invokes or does not invoke the original function call, based on the inspection results.

Reference is now made to FIG. 5, which is a simplified flowchart of a method for protecting a computer from dynamically generated malicious executable code, whereby

16

the gateway computer itself performs the code inspection, in accordance with a preferred embodiment of the present invention. The leftmost column indicates steps performed by a gateway computer, such as gateway computer 405; and the rightmost column indicates steps performed by a client computer, such as client computer 410.

The method illustrated in FIG. 5 is similar to that of FIG. 3, where steps 340-380 performed by the security computer in FIG. 3 are performed by the gateway computer in FIG. 5. At step 500 the gateway computer receives content from a network, the content intended for delivery to the client computer. At step 505 the gateway computer scans the content for the presence of function calls. At step 510 the gateway computer branches. If function calls within the content were detected at step 505, then at step 515 the gateway computer modifies the content by replacing original function calls of the form (1) with corresponding substitute function calls of the form (2). Otherwise, if function calls were not detected at step 505, then the gateway computer skips step 515. At step 520 the gateway computer transmits the content, which may have been modified at step 515, to the client computer.

At step 525 the client computer receives the content from the gateway computer, and at step 530 the client computer begins processing the content. While processing the content, the client computer invokes a substitute function call of the form (2) at step 535. The substitute function, being of the form listed on TABLE I, instructs the client computer to transmit the function input and a client computer identifier to the gateway computer for inspection. At step 540 the client computer transmits the input and the identifier to the gateway computer, and suspends processing of the content pending a reply from the gateway computer.

At step 545 the gateway computer receives the input and the client identifier from the client computer, and loops back to step 505 to scan the input for the presence of function calls. At step 510 the gateway computer branches. If function calls within the Input were detected at step 505, then the gateway computer modifies the input at step 515, by replacing function calls of the form (1) with corresponding function calls of the form (2). Otherwise, if function calls were not detected at step 505, then the gateway computer skips step 515.

The gateway computer then proceeds to step 550, and scans the input, which may have been modified at step 515, to identify potentially malicious operations within the input. The potentially malicious operations identified form a security profile for the input.

At step 555 the gateway computer retrieves a security policy for the client computer from a database of security policies. At step 560 the gateway computer compares the input's security profile with the client computer's security policy to determine whether or not the security profile violates the security policy. At step 565 the gateway computer branches. If the results of step 560 indicate that the input security profile does not violate the client computer security policy, then it is safe for the client to invoke the original function call, and an indicator of the inspection results is set to true at step 570. Otherwise, the indicator is set to false at step 575. At step 580 the gateway computer returns the indicator to the client computer. The gateway computer may also return the modified input, as modified at step 515, to the client computer.

At step 585 the client computer receives the reply back from the gateway computer and resumes processing of the content, which processing had been suspended after step 540. At step 590 the client computer branches. If the indicator was set to true by the gateway computer at step 570, then the client computer invokes the original function call (1). If the gateway

US 8,141,154 B2

17

computer had modified the input at step 515, then preferably the client computer uses the modified input instead of the original input when invoking the original function call. Otherwise, if the indicator was set to false by the gateway computer at step 575, then the client computer skips step 595. The client computer then loops back to step 530 to continue processing of the content.

Having read the above disclosure, it will be appreciated by those skilled in the art that the present invention can be used to provide protection to computers against both statically and dynamically generated malicious code. Moreover, such protection may be afforded by a security computer that is remote from the computers being protected, thus adding another layer of security to methods and systems that embody the present invention.

In reading the above description, persons skilled in the art will realize that there are many apparent variations that can be applied to the methods and systems described. Thus it may be appreciated that the present invention applies to a variety of computing devices, including mobile devices with wireless Internet connections such as laptops, PDAs and cell phones.

In the foregoing specification, the invention has been described with reference to specific exemplary embodiments thereof. It will, however, be evident that various modifications and changes may be made to the specific exemplary embodiments without departing from the broader spirit and scope of the invention as set forth in the appended claims. Accordingly, the specification and drawings are to be regarded in an illustrative rather than a restrictive sense.

What is claimed is:

1. A system for protecting a computer from dynamically generated malicious content, comprising:

a content processor (i) for processing content received over a network, the content including a call to a first function, and the call including an input, and (ii) for invoking a second function with the input, only if a security computer indicates that such invocation is safe;

a transmitter for transmitting the input to the security computer for inspection, when the first function is invoked; and

a receiver for receiving an indicator from the security computer whether it is safe to invoke the second function with the input.

2. The system of claim 1 wherein said content processor (i) suspends processing of the content after said transmitter transmits the input to the security computer, and (ii) resumes processing of the content after said receiver receives the indicator from the security computer.

3. The system of claim 1 wherein the input is dynamically generated by said content processor prior to being transmitted by said transmitter.

4. A non-transitory computer-readable storage medium storing program code for causing a computing device to:

process content received over a network, the content including a call to a first function, and the call including an input;

transmit the input for inspection, when the first function is invoked, and suspend processing of the content;

receive an indicator of whether it is safe to invoke a second function with the input; and

18

resume processing of the content after receiving the indicator, and invoke the second function with the input only if the indicator indicates that such invocation is safe.

5. The non-transitory computer-readable storage medium of claim 4 wherein the program code causes the computer device to dynamically generate the input prior to transmitting the input for inspection.

6. A system for protecting a computer from dynamically generated malicious content, comprising:

a content processor (i) for processing content received over a network, the content including a call to a first function, and the first function including an input variable, and (ii) for calling a second function with a modified input variable;

a transmitter for transmitting the input variable to a security computer for inspection, when the first function is called; and

a receiver for receiving the modified input variable from the security computer,

wherein the modified input variable is obtained by modifying the input variable if the security computer determines that calling a function with the input variable may not be safe.

7. The system of claim 6 wherein said content processor (i) suspends processing of the content after said transmitter transmits the input variable to the security computer, and (ii) resumes processing of the content after said receiver receives the modified input variable from the security computer.

8. The system of claim 6 wherein the input variable is dynamically generated by said content processor prior to being transmitted by said transmitter.

9. The system of claim 6 wherein the input variable includes a call to an additional function, and wherein the modified input variable includes a call to a modified additional function instead of the call to the additional function.

10. A non-transitory computer-readable storage medium storing program code for causing a computing device to:

process content received over a network, the content including a call to a first function, and the first function including an input variable;

transmit the input variable for inspection, when the first function is called, and suspend processing of the content;

receive a modified input variable; and

resume processing of the content after receiving the modified input variable, and calling a second function with the modified input variable,

wherein the modified input variable is obtained by modifying the input variable if the inspection of the input variable indicates that calling a function with the input variable may not be safe.

11. The non-transitory computer-readable storage medium of claim 10 wherein the program code causes the computer device to dynamically generate the input variable prior to transmitting the input variable for inspection.

12. The non-transitory computer-readable storage medium of claim 10 wherein the input variable includes a call to an additional function, and wherein the modified input variable includes a call to a modified additional function instead of the call to the additional function.

\* \* \* \* \*

UNITED STATES PATENT AND TRADEMARK OFFICE  
**CERTIFICATE OF CORRECTION**

PATENT NO. : 8,141,154 B2  
APPLICATION NO. : 12/814584  
DATED : March 20, 2012  
INVENTOR(S) : David Gruzman et al.

Page 1 of 1

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

On the Title Page, insert Item (63):

-- Related U.S. Application Data --

-- (63) Divisional of application no. 11/298,475, filed on Dec. 12, 2005, Now Pat. No. 7,757,289. --


In the Specification

In Column 1, add the following heading and paragraph directly below the title of the invention:

-- CROSS-REFERENCE TO RELATED APPLICATIONS --

-- This application is a divisional of and claims priority to U.S. Patent Application Serial No. 11/298,475, filed December 12, 2005, entitled "System and Method For Inspecting Dynamically Generated Executable Code," now U.S. Patent No. 7,757,289. --

Signed and Sealed this  
Twenty-fifth Day of February, 2014



Michelle K. Lee  
*Deputy Director of the United States Patent and Trademark Office*

# **EXHIBIT 6**

[Trials@uspto.gov](mailto:Trials@uspto.gov)  
571-272-7822

Paper 49  
Entered: March 16, 2017

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

PALO ALTO NETWORKS, INC.,  
Petitioner,

v.

FINJAN, INC.,  
Patent Owner.

---

Case IPR2015-01974<sup>1</sup>  
Patent 7,647,633 B2

---

Before, THOMAS L. GIANNETTI, MIRIAM L. QUINN, and  
PATRICK M. BOUCHER *Administrative Patent Judges*.

QUINN, *Administrative Patent Judge*.

FINAL WRITTEN DECISION  
*35 U.S.C. § 318(a) and 37 C.F.R. § 42.73*

---

<sup>1</sup> Case IPR2016-00480 (filed by Blue Coat Systems, Inc.) has been joined with this proceeding.

IPR2015-01974  
Patent 7,647,633 B2

Palo Alto Networks, Inc. and Blue Coat Systems, Inc. (“Petitioner”) each filed a Petition to institute *inter partes* review of claims 1–4, 6–8, 13, 14, 19, 28, and 34 of U.S. Patent No. 7,647,633 B2 (“the ’633 patent”) pursuant to 35 U.S.C. § 311–319. IPR2015-01974, Paper 1 (“Pet.”); IPR2016-00480, Paper 3. Finjan, Inc. (“Patent Owner”) filed a Preliminary Response in both proceedings. IPR2015-01974, Paper 6; IPR2016-00480, Paper 8. Upon consideration of the information submitted by the parties at the preliminary stage, we instituted trial only as to claims 14 and 19 of the ’633 patent. Paper 7 (“Dec.”). We also granted Blue Coat Systems, Inc.’s motion requesting joinder of IPR2016-00480 with this proceeding. Paper 17. We terminated Case IPR2016-00480, and ordered consolidation of all Petitioner filings in this proceeding. *Id.* at 10.

During trial, Patent Owner filed a Patent Owner Response (Paper 22 (“PO Resp.”)); and Petitioner filed a Reply (Paper 31 (“Reply”)). Both parties filed Motions to Exclude, Oppositions, and Replies in connection with those Motions. Papers 35, 36, 39, 40, 42, and 43. We held oral argument on January 5, 2017. Paper 48 (“Tr.”).

We have jurisdiction under 35 U.S.C. § 6. This Final Written Decision is issued pursuant to 35 U.S.C. § 318(a). For the reasons discussed herein, and in view of the record in this trial, we determine that Petitioner has not shown by a preponderance of the evidence that claims 14 and 19 of the ’633 patent are unpatentable.

IPR2015-01974  
Patent 7,647,633 B2

## I. BACKGROUND

### A. RELATED MATTERS

Petitioner identifies the '633 patent as the subject of various district court cases filed in the U.S. District Court for the Northern District of California (Case Nos. 3-14-cv-04908, 13-cv-03133, 13-cv-03999, 5-13-cv-04398, 13-cv-05808, and 5-15-cv-01353). Pet. 2. Petitioner also states that petitions for *inter partes* review have been filed regarding other patents assigned to Patent Owner. *Id.*

The '633 patent is also the subject of two *ex parte* reexamination proceedings with Control Nos. 90/013,016 and 90/013,652. Paper 46 (Patent Owner updated mandatory notice pursuant to 37 C.F.R. § 42.8). Neither of these *ex parte* reexamination proceedings involves the claims-at-issue in this *inter partes* review.

### B. THE '633 PATENT (EX. 1001)

The '633 patent relates to a system and a method for protecting network-connectable devices from undesirable downloadable operation. Ex. 1001, 1:30–33. The patent describes that “Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others).” *Id.* at 1:60–63. Protecting against only some distributable components does not protect against application programs, Trojan horses, or zip or meta files, which are other types of Downloadable information. *Id.* at 1:63–2:2. The '633 patent “enables more reliable protection.” *Id.* at 2:27–28. According to the Summary of the Invention,

IPR2015-01974

Patent 7,647,633 B2

In one aspect, embodiments of the invention provide for determining, within one or more network “servers” (e.g. firewalls, resources, gateways, email relays or other devices/processes that are capable of receiving-and-transferring a Downloadable) whether received information includes executable code (and is a “Downloadable”). Embodiments also provide for delivering static, configurable and/or extensible remotely operable protection policies to a Downloadable-destination, more typically as a sandboxed package including the mobile protection code, downloadable policies and one or more received Downloadables. Further client-based or remote protection code/policies can also be utilized in a distributed manner. Embodiments also provide for causing the mobile protection code to be executed within a Downloadable-destination in a manner that enables various Downloadable operations to be detected, intercepted or further responded to via protection operations. Additional server/information-destination device security or other protection is also enabled, among still further aspects.

*Id.* at 2:39–57.

### C. CHALLENGED CLAIMS

Challenged claims 14 and 19 are reproduced below.

14. A computer program product, comprising a computer usable medium having a computer readable program code therein, the computer readable program code adapted to be executed for computer security, the method comprising:

    providing a system, wherein the system comprises distinct software modules, and wherein the distinct software modules comprise an information re-communicator and a mobile code executor;

    receiving, at the information re-communicator, downloadable-information including executable code; and

    causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable



IPR2015-01974  
 Patent 7,647,633 B2

code at the destination, if attempted, will be processed by the mobile protection code.

19. The method of claim 14, wherein the re-communicator is at least one of a firewall and a network server.

*Id.* at 21:58–22:5, 22:15–16.

#### D. INSTITUTED GROUNDS

We instituted *inter partes* review of claims 14 and 19 based on the following grounds (Dec. 13–16):

Reference(s)	Basis	Claims
Shin <sup>2</sup>	§ 103	14 and 19
Poison Java <sup>3</sup> and Brown <sup>4</sup>	§ 103	14 and 19

Petitioner supports its contentions of unpatentability with a declaration from Dr. Aviel Rubin. Ex. 1002. Patent Owner supports its contentions of patentability with a declaration from Dr. Michael Goodrich. Ex. 2019. Patent Owner also proffers as support a declaration from Dr. Harry Bims (Ex. 2020) and Michael Kim (Ex. 2021). The cross-examinations of Drs. Rubin, Goodrich, and Bim are in the record as Exhibits 2022, 1097, and 1098, respectively.

<sup>2</sup> Insik Shin, et al., *Java Bytecode Modification and Applet Security* (Technical Report, Computer Science Dept., Stanford University, 1998), <https://web.archive.org/web/19980418130342/http://www-cs-students.stanford.edu/~ishin/reserach.html> (Ex. 1009) (“Shin”).

<sup>3</sup> Eva Chen, *Poison Java*, IEEE SPECTRUM, August 1999 at 38 (Ex. 1004).

<sup>4</sup> Mark W. Brown, et al., SPECIAL EDITION USING NETSCAPE 3, (Que Corp. 1996) (Ex. 1041) (“Brown”).

IPR2015-01974  
Patent 7,647,633 B2

## II. ANALYSIS

### A. CLAIM INTERPRETATION

In an *inter partes* review, claim terms in an unexpired patent are interpreted according to their broadest reasonable construction in light of the specification of the patent in which they appear. 37 C.F.R. § 42.100(b); *Cuozzo Speed Techs., LLC v. Lee*, 136 S. Ct. 2131, 2142–46 (2016). Consistent with that standard, claim terms also are given their ordinary and customary meaning, as would be understood by one of ordinary skill in the art in the context of the entire disclosure. *See In re Translogic Tech., Inc.*, 504 F.3d 1249, 1257 (Fed. Cir. 2007). Although it is improper to read a limitation from the specification into the claims, *In re Van Geuns*, 988 F.2d 1181, 1184 (Fed. Cir. 1993), claims still must be read in view of the specification of which they are a part. *Microsoft Corp. v. Multi-Tech Sys., Inc.*, 357 F.3d 1340, 1347 (Fed. Cir. 2004).

In our Decision on Institution, we did not construe expressly any claim terms. During trial, Patent Owner proposed a construction for the following phrase: “causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code.” PO Resp. 12–13. Patent Owner urges the Board to construe the phrase according to its plain and ordinary meaning, with the explanation that “the mobile protection code was communicated to the downloadable-information destination without modifying the executable code.” *Id.* Patent Owner argues that a District Court has construed the phrase and concluded that “[i]t is clear from the specification that the MPC [(mobile protection code)], does not modify

IPR2015-01974  
Patent 7,647,633 B2

the executable code.” *Id.* at 13 (citing Ex. 1036, 9). In particular, Patent Owner points out the District Court’s reliance on a stated advantage of the ’633 patent invention: protecting against malicious operations without modifying the mobile code and the specification’s statements distinguishing the invention from the prior art on this basis. *Id.* at 13–14 (citing Ex. 1001 at 4:12–16, 10:39–44). Patent Owner also relies on various expert declarations submitted in connection with the District Court litigation. *Id.* at 15–16.

Petitioner counters that the Board should construe the phrase under the broadest reasonable interpretation, and that under that interpretation the claim language should be given its plain and ordinary meaning. Reply 2–3. Petitioner, however, argues that the “without modification” aspect of Patent Owner’s proposed construction is supported by neither the claim language nor the specification. *Id.* at 3–5. Moreover, Petitioner argues that a Finjan expert, Dr. Harry Bims, although opining on issues of secondary considerations of nonobviousness (Ex. 2020), confirms that Patent Owner’s proposed construction is at odds with the plain and ordinary meaning of the phrase. *Id.* at 5–7.

Claim 14 is directed to a computer program product adapted to execute three steps. The first step is directed to providing a system of software modules, one of which is a re-communicator, the other being a mobile code executor. In the second step, the re-communicator receives “downloadable-information including executable code.” In the third and final step, the executable code is processed at the destination by the mobile protection code, if one or more operations in that code are attempted. The issue before us is whether the executable code at the destination could be a

IPR2015-01974  
Patent 7,647,633 B2

modified code, in comparison to the executable code that the re-communicator received. Having reviewed the evidence and arguments proffered by both parties on this issue, we determine that the claimed “executable code” is not modified prior to being processed by the mobile protection code at the destination.

First, we note that the claim language, itself, is silent regarding the modified or unmodified status of the “executable code” at the destination. The omission, however, is not dispositive because the language of claim 14 focuses on the functionality of the mobile protection code vis-à-vis the executable code. For example, the claim focuses on execution of the mobile protection code at a downloadable-information destination to process the “one or more operations of the executable code at the destination, if attempted.” Ex. 1001, 22:1–4. The claim language, therefore, implies that the protection from malicious code is provided by the mobile protection code processing the attempted operations of the downloaded executable code. Considering the objective of the invention is to protect a computer from “malicious” operations, we understand claim 14 to require processing the executable code which may result in malicious operations, as received, at the intended computer destination via the mobile protection code. In other words, the claim language may be understood to require that the “executable code” received at the re-communicator is the same code whose “one or more operations” are being processed at the destination by the mobile protection code. *See* PO Resp. 14–15 (citing Ex. 2019, the Goodrich Declaration, ¶ 36).

Petitioner argues that the broader interpretation consistent with a plain and ordinary meaning may not require that the entire “executable code”

IPR2015-01974

Patent 7,647,633 B2

received at the re-communicator is the same executable code processed at the destination. Reply 3. At oral argument, Petitioner agreed that the claimed “executable code” at the destination recites an antecedent that refers back to the received executable code, and is the same code. Tr. 9:8–14. The claim language, according to Petitioner, although referring to the same code, does not resolve the issue whether some of the executable code processed at the destination may be modified. *Id.* at 9:15–25; 10:22–11:10 (Petitioner arguing that the claimed “one or more operations” of the executable code at the destination “could refer to either modified or unmodified.”). We agree with Petitioner that although the claim language refers to the executable code being received at the destination, the question of whether operations of the code may be modified is unanswered by the claim language alone. Our inquiry, however, is not based on the claim language in a vacuum, as the appropriate meaning of the claim language emerges when viewing the claim in the context of the specification. *See Innova/Pure Water, Inc. v. Safari Water Filtration Sys., Inc.*, 381 F.3d 1111, 1116 (Fed. Cir. 2004) (stating that the specification provides context and may be relied on to understand the meaning of the claim).

We acknowledge that the specification does not define “executable code” or any of the words of the disputed claim phrase. With regard to the executable code and its processing, we note that the background of the invention fairly describes it as “program code”: “It is observed by this inventor for example, that Downloadable information comprising program code can include distributable components . . . .” Ex.1001, 1:60–63. There is also a reference in the specification to “application programs,” and programs in various forms: “[Downloadable information] can also include,

IPR2015-01974  
Patent 7,647,633 B2

for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others.” *Id.* 1:63–66; *see also* 2:29–33 (“remotely operable code that is protectable against can include downloadable application programs, Trojan horses and program code groupings, as well as software ‘components,’ such as Java™ applets, ActiveX™ controls, JavaScript™/Visual Basic scripts, add-ins, etc., among others.”). From these descriptions, we conclude that it would be apparent to a person of ordinary skill that the claimed “executable code” was meant to include programs or computer instructions that are executed by a computer. The executable nature of the “executable code” is essential to the nature of the problem addressed by the ’633 patent claims, i.e., virus protection. According to the background of the invention, a virus is “potentially system-fatal or otherwise damaging *computer code*.” *Id.* at 1:40–44 (emphasis added).

In describing the problems with the virus protection systems in the prior art, the inventors recognized known techniques for protecting a sub-set of executable code, namely only Java applets and ActiveX controls. *Id.* at 1:65–2:2. But more importantly, the inventors disclosed that this prior art, specifically U.S. Patent No. 5,983,348 to Shuang, Ji (Ex. 2006, “Shuang-Ji”), was resource intensive, used high bandwidth static and operational analysis of the downloadable content, and modified the Downloadable component. *Id.* at 2:2–4.<sup>5</sup> According to the inventors,

---

<sup>5</sup> Shuang-Ji is directed to scanning an applet at a server to identify problematic instructions, which are then instrumented by inserting special code before and after each problematic instruction. Ex. 2006, 3:16–31. Alternatively, the problematic instruction may be replaced. *Id.*

IPR2015-01974  
Patent 7,647,633 B2

Shuang-Ji also lacked the efficiency and flexibility afforded by the invention. *Id.* at 2:10–13. The embodiments of the '633 patent all involve receiving at, or sending to, a downloadable destination, a mobile protection code and the downloadable information that includes the executable code. *Id.* at 2:39–3:62.

The mobile protection code is installed at the user device (downloadable destination) and monitors resource access attempts by the executable code in the downloadable information, such that those attempts may be dealt with in a variety of ways. *Id.* at 3:63–4:10. One of those ways includes the option of “modifying the Downloadable operation.” *Id.* However, it is important to note that the mobile protection code, being executed at the client, is not described as modifying the executable code itself before or after the executable code is received at the destination. The relevance of this passage is that, notwithstanding the threat of the malicious operations, the destination executes the instructions in the executable code. The mobile protection code intercepts the potentially malicious operations resulting from the execution of those instructions. In sum, all the embodiments described in the '633 patent defer the processing of the malicious operations to the destination, with one of those processing options being the modification of the operation. However, no embodiment describes modification of the executable code itself.

---

Nevertheless, the instrumentation occurs at a server, and the instrumented applet is downloaded to the client. *Id.* at 3:32–35. In other words, the applet, or executable code, is modified at the server before delivery to the client.

IPR2015-01974  
Patent 7,647,633 B2

In addition to providing the above descriptions of the executable code, we find that the specification expressly describes an advantage of non-modification of the executable code as follows:

Advantageously, systems and methods according to embodiments of the invention enable potentially damaging, undesirable or otherwise malicious operations by even unknown mobile code to be detected, prevented, modified and/or otherwise protected against *without modifying the mobile code*. Such protection is further enabled in a manner that is capable of minimizing server and client resource requirements, does not require pre-installation of security code within a Downloadable-destination, and provides for client specific or generic and readily updateable security measures to be flexibly and efficiently implemented.

Ex. 1001, 4:11–21 (emphasis added). This passage uses the phrase “mobile code” not “executable code.” We find, however, that the “mobile code” being referred to here is the code that may result in the potentially malicious operations sought to be prevented. *Id.* (referring to enabling detection of “potentially damaging, undesirable or otherwise malicious operations by even unknown *mobile code*”) (emphasis added). Thus, this passage is consistent with the statements regarding the difference between Shuang-Ji’s modification of the downloadable component, i.e. executable code. *See e.g., id.* at 1:63–2:6 (referring to Shuang-Ji requiring modification of the “Downloadable component,” which is a Java applet or ActiveX controls components included in the Downloadable information). In sum, while the prior art modifies the downloadable component, the claimed invention does not. And this is an advantage touted by the inventors in describing that communicating the mobile protection code separate from the downloadable that includes executable code is more accurate and “far less resource



IPR2015-01974  
Patent 7,647,633 B2

intensive than, for example, performing content and operation scanning, modifying a Downloadable, or providing completely Downloadable-destination based security.” *Id.* at 10:39–45.

Notwithstanding the specification passages analyzed above, Petitioner asserts that embodiments of the ’633 patent specification disclose modification of the executable code. Reply 4–5. Specifically, Petitioner points out the following passage: “[T]he IAT [(API Import Address Table)] can be modified so that any call to an API can be redirected to a function within the MPC [mobile protection code].” *Id.* at 4 (citing Ex. 1001, 17:51–53); Tr. 12:11–13:8. As further evidence of modification of executable code embodiments, Petitioner argues the description of compression, encryption, or encoding of executable code discloses modified executable code. *Id.* (citing Ex. 1001, 13:29–30). As support, Petitioner alleges that Patent Owner’s expert agrees that compression comprises modification. *Id.* at 5 (citing Ex. 1100, 173:22–174:16).

We are not persuaded by Petitioner’s arguments. First, we find that the embodiment concerning the IAT table modification does not disclose modification of executable code. The embodiment Petitioner refers to describes the functionality of the mobile protection code after installation at the destination. Ex. 1001, 17:13–19 (describing an embodiment shown in Figure 7a of the client receiving a sandbox package and initiating a mobile code installer, which initiates the mobile protection code); 17:28–42 (describing that the mobile protection code intercepts an operation, and, as one example, limits access to resources, such as address space, in the event the operation is malicious). As explained above, the specification does describe modification of operations performed at the destination when the

IPR2015-01974

Patent 7,647,633 B2

mobile protection code detects malicious code. But the modifications are (1) to the operation resulting from execution of the executable code; and (2) at the destination, after the executable code has been received at the destination. The IAT embodiment Petitioner points to is one such embodiment, where the mobile protection code modifies an operation at the destination. For instance, the mobile protection code loads in memory the executable code, shown in Figure 7a as “XEQ” 343. *Id.* at 17:43–47. The IAT, illustrated as “API” 731 in Figure 7b, is loaded in memory also so that the mobile protection code can access the table and divert the access attempts of the executable code to an analyzer. *See id.* at 17:58–63 (stating that initial access to the IAT provides for diverting, evaluating, and responding to attempts by the downloadable to utilize system APIs); 18:20–24 (describing that a resource access diverter modifies that IAT entries causing access attempts to be diverted to the resource access analyzer). In short, the ’633 patent specification describes modification of the IAT after the executable code in the downloadable has been received and as part of the processing of the malicious operations by the mobile protection code.

Given the above disclosure, arguments by the parties regarding whether the IAT is executable and whether the IAT is part of the downloadable are not germane to our determination. *See* Tr. 14:13–15:5 (Petitioner asserting that the IAT is part of the downloadable); 43:6–13 (Patent Owner arguing that the IAT is not executable code and the modification occurs at the client once the mobile protection code has access to it). As stated above, we find that the IAT disclosure in the ’633 patent describes the claimed processing by the mobile protection code of the

IPR2015-01974  
Patent 7,647,633 B2

attempted operations of the executable code. This disclosure does not support Petitioner's assertion that the specification contemplates modification of the executable code prior to sending it to the client for processing there.<sup>6</sup> Indeed, such a disclosure would be contrary to the stated advantages and distinctions discussed above of mobile protection code that detects, prevents, and modifies malicious operations of executable code, without modifying such executable code.

Finally, we address whether the alleged disclosure of compression, encryption, and encoding supports Petitioner's contention that the '633 patent specification discloses modification of executable code. Reply 4–5 (citing Ex. 1001 at 13:29–30). Patent Owner contends that the disclosure of compression, encryption, and encoding does not disclose modification of the executable code. Tr. 39:3–40:15. One of the confusing aspects of this case is that evidence proffered by Petitioner as supporting its claim construction position is taken from Patent Owner's evidence submitted during the District Court litigation, before the Court there ruled against Patent Owner on this issue. For instance, Petitioner points to Dr. Rubin's testimony during cross-examination as supporting the contention that compression constitutes modification. Reply 4–5 (citing Ex. 2022, 108:4–11). Upon close inspection, however, we understand that Dr. Rubin read into the record portions of Finjan's brief on claim construction submitted in the District Court litigation. Ex. 2022, 107:4–108:11 (referring to Plaintiff Finjan, Inc.'s

---

<sup>6</sup> We also disagree with Petitioner that Dr. Goodrich's testimony supports Petitioner's position on this issue. Dr. Goodrich testified that although the '633 patent describes modifying the IAT, that modification does not constitute modifying the downloadable. Ex. 1097, 25:7–27:19.

IPR2015-01974  
Patent 7,647,633 B2

Opening Claim Construction Brief (Exhibit 1039), page 17 (Petitioner's pagination in the footer of the exhibit), sentence starting at line 15).<sup>7</sup> The reality here is that Patent Owner was not successful in persuading the District Court that the disputed '633 patent claims (including claim 14) are broad enough to encompass communicating both un-modified and modified executable code to a client. *See* Ex. 1036, 8–15; Tr. 41:10–42:14 (Patent Owner explaining that Finjan adopted the district court's construction, did not object to it even though it was not the construction it proffered, and that it has not raised this issue on appeal). The intrinsic record is sufficiently clear that we have no need to rely on expert testimony presented in District Court.<sup>8</sup> And the intrinsic record does not show that operations of compression, encryption, and encoding constitute a modification of executable code. The '633 patent discloses “restoring” the downloadable, not modifying it. Ex. 1001, 13:29–32. That is, if a downloadable is received in a compressed format, it is decompressed to detect whether there is executable code. *Id.* The downloadable then is *restored* to its compressed format for transmission and subsequent processing at the client. *Id.* This is not teaching modification of executable code because the code itself remains unchanged.

To conclude, our determination is based on the plain and ordinary meaning of the claims within the context of the specification. First,

---

<sup>7</sup> Similarly, Petitioner relies on testimony from the deposition of Finjan's expert in the District Court litigation, Dr. Nenad Medvidovic, Exhibit 1100. Reply 5 (citing Ex. 1100).

<sup>8</sup> We do not rely on expert testimony also because extrinsic evidence may not be used to contradict claim meaning that is unambiguous in light of the intrinsic record. *Summit 6, LLC v. Samsung Elecs. Co.*, 802 F.3d 1283, 1290 (Fed. Cir. 2015).

IPR2015-01974

Patent 7,647,633 B2

according to the entire disclosure of the '633 patent specification the executable code is received at the destination without modification. *See Vitronics Corp. v. Conceptronic, Inc.*, 90 F.3d 1576, 1582 (Fed. Cir. 1996) (the specification is “the single best guide to the meaning of a disputed term,” and “acts as a dictionary when it expressly defines terms used in the claims or when it defines terms by implication.”). “The specification may define claim terms ‘by implication’ such that the meaning may be ‘found in or ascertained by a reading of the patent documents.’” *Bell Atl. Network Servs., Inc. v. Covad Communications Group, Inc.*, 262 F.3d 1258, 1268 (Fed.Cir.2001) (quoting *Vitronics*, 90 F.3d at 1582, 1584 n. 6). We also confirm the scope of the “executable code” in light of the advantage of the invention, e.g., to protect against malicious code without modifying the executable code, thereby minimizing resources. *See Toro Co. v. White Consol. Indus., Inc.*, 199 F.3d 1295, 1301 (Fed. Cir. 1999) (construing a ring as permanently attached to the cover because all embodiments read together suggest this relationship and the specification describes the advantages of the unitary structure). Although the specification describes additional advantages, such as not requiring pre-installation of security code within the destination, the claims need not encompass all stated advantages. *See Golight, Inc. v. Wal-Mart Stores, Inc.*, 355 F.3d 1327, 1331 (Fed. Cir. 2004) (“[P]atentees [are] not required to include within each of their claims all of [the] advantages or features described as significant or important in the written description.”). Finally, all the embodiments are consistent with and support the inventors’ characterization of the problem solved: avoiding the resource intensive modification by communicating to the destination the mobile protection code and the detected executable code. *See Ex. 1001*,

IPR2015-01974  
Patent 7,647,633 B2

10:39–45; 1:63–3:6; *see also O.I. Corp. v. Tekmar Co.*, 115 F.3d 1576, 1581 (Fed. Cir. 1997) (relying on the consistent disclosure of structures as being either non-smooth or conical and the statements distinguishing prior art where structures were smooth). In light of the disclosure, the plain and ordinary meaning of the disputed claim language emerges: the executable code whose operations are processed by the mobile protection code at the destination is the same as the executable code received, i.e., it undergoes no modification.

In sum, we determine that the “executable code” at the destination is not modified.

#### B. PRINTED PUBLICATION ISSUES

The three asserted references are non-patent documents. Shin (Ex. 1009) is an article that Petitioner alleges was publicly available via the Internet in 1998. Pet. 29. Poison Java (Ex. 1004) is an article that Petitioner asserts was published in the August 1999 issue of the IEEE Spectrum magazine. Pet. 25. And Brown (Ex. 1041) is a guide to the Netscape browser and appears to be a book that Petitioner asserts was published in 1996. Pet. 27. Patent Owner argues that Petitioner did not meet its burden in showing that the three references are printed publications. Based on our review of the evidence of record, we determine that Shin, Poison Java, and Brown are printed publications as described further below.

##### 1. *Shin*

On the face of this article, we note that there are no indicia of publication or dissemination. *See* Ex. 1009. Petitioner, however, has provided other evidence establishing that Shin was publicly available no later than April 1998. A printout of the website directed to Mr. Shin’s

IPR2015-01974  
Patent 7,647,633 B2

research at Stanford, and archived by archive.org, shows that the article's title was listed and that the article was publicly available through the Internet as of April 14, 1998. Ex. 1095. As explained in the affidavit proffered by Christopher Butler, Office Manager of Internet Archive, the URL format for captured website, "19980418130342/http://www-cs-students.stanford.edu/~ishin/research.html," establishes that the website posting the abstract and the title of the article with a link to the article itself were archived on April 18, 1998 at 13:03:42. *Id.* The article was available as a PDF file, downloadable from the same page. *Id.* (identifying in a cover page the URL showing that Ex. 1009 was available from www-cs-students.stanford.edu/~ishin/paper.ps, as of April 18, 1998). We find this evidence credible and undisputed.

Finally, there is also undisputed evidence that Dr. Rubin read Shin in 1998. Ex. 2022, 70:20–71:13. Accordingly, we find that Shin was publicly available and sufficiently accessible online to persons interested in applet security no later than April 14, 1998.

## 2. *Poison Java*

On the face of this reference, we note that Poison Java contains multiple indicia that it is part of a periodical, IEEE Spectrum, published in August 1999. Ex. 1004. This fact is confirmed by the declaration and testimony of Gerard P. Grenier, Senior Director of Publishing Technologies and custodian of certain records at IEEE. Exs. 1005, 2023. In particular, we note that Mr. Grenier testified that, although the electronic copies of Poison Java are now maintained in IEEE Xplore, the article itself was published in Volume 36, Issue 8 of IEEE Spectrum in August 1999. Ex. 1005. The printouts attached to Mr. Grenier's declaration show that the database IEEE

IPR2015-01974  
Patent 7,647,633 B2

Xplore deems Poison Java officially published in August 1999. *Id.* at Ex. A; Ex. 2023, 29:19–20, 32:7–34:4 (testifying that IEEE Spectrum is a flagship publication of IEEE mailed to members every month and that it is publishing practice to mail by no later than the first day of the month published on the cover).

Patent Owner makes much about the fact that the IEEE Xplore database would not have been available until June 2000. PO Resp. 23–24. This argument, however, ignores that the article was not *only* available as a download in the IEEE database. The evidence shows that the article was included in a monthly publication of the magazine IEEE Spectrum and, according to the publishing practices of IEEE, the magazine issue containing Poison Java would have been received by subscribers by August 1999. Accordingly, we find that Poison Java is a printed publication, disseminated to subscribers of the IEEE Spectrum in August 1999.

### 3. *Brown*

Petitioner asserts that Brown (Ex. 1041) was published in 1996, as confirmed by a declaration of one of the book's authors and the Internet Archive. Exs. 1082, 1092. Peter Kent, one of the authors, testifies that he received a copy of the book shortly after publication in December 1996, and that the publisher sent periodic sales updates beginning in 1996. Ex. 1082 ¶¶ 3–6. An affidavit from Christopher Butler, of Internet Archive, provides evidence that a website listing books by Que, the publisher of Brown, listed the book for sale as of the date of the archived page, December 19, 1996. *See* 1092 (showing capture of webpage <http://www.mcp.com/que/et/se-net3/>). On cross-examination, Mr. Kent testified that the book was actually published on September 1996, not



IPR2015-01974  
Patent 7,647,633 B2

December 1996. Ex. 2024, 20:20–22:25. The testimony leads to the conclusion that the book was printed in 1996, notwithstanding the misunderstanding regarding whether September or December was the first month of publication.

Furthermore, we find persuasive the evidence from the Internet Archive that Brown was available for sale on the Internet as of December 16, 1996. Notwithstanding Patent Owner’s argument that the link to buy the book has not been shown to work, we find persuasive that Brown was available for purchase over the Internet, which was searchable using a browser. *See* Ex. 2024, 33:12–17. In summary, the record evidence on the whole persuades us that Brown was published and offered for sale in the United States in 1999. We are persuaded that by December 1999, anyone with interest in understanding the use of the Netscape browser would have been able to buy it. *See In re Wyer*, 655 F.2d 221, 227, (CCPA 1981) (“Accordingly, whether information is printed, handwritten, or on microfilm or a magnetic disc or tape, etc., the one who wishes to characterize the information, in whatever form it may be, as a ‘printed publication’ ... should produce sufficient proof of its dissemination or that it has otherwise been available and accessible to persons concerned with the art to which the document relates and thus most likely to avail themselves of its contents.”) (citations omitted).

### C. PRINCIPLES OF LAW

A claim is unpatentable under 35 U.S.C. § 103(a) if the differences between the claimed subject matter and the prior art are such that the subject matter, as a whole, would have been obvious at the time the invention was

IPR2015-01974  
Patent 7,647,633 B2

made to a person having ordinary skill in the art to which said subject matter pertains. *KSR Int'l Co. v. Teleflex Inc.*, 550 U.S. 398, 406 (2007). The question of obviousness is resolved on the basis of underlying factual determinations including: (1) the scope and content of the prior art; (2) any differences between the claimed subject matter and the prior art; (3) the level of ordinary skill in the art; and (4) objective evidence of nonobviousness. *Graham v. John Deere Co.*, 383 U.S. 1, 17–18 (1966).

#### D. THE LEVEL OF SKILL IN THE ART

In determining the level of ordinary skill in the art at the time of the invention, we note that various factors may be considered, including “type of problems encountered in the art; prior art solutions to those problems; rapidity with which innovations are made; sophistication of the technology; and educational level of active workers in the field.” *In re GPAC, Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995) (citing *Custom Accessories, Inc. v. Jeffrey-Allan Indus., Inc.*, 807 F.2d 955, 962 (Fed. Cir. 1986)).

Petitioner asserts, through its expert, Dr. Aviel Rubin, that the “relevant technology field for the ’154 patent is security programs, including content scanners for program code.” Pet. 24; Ex. 1002 ¶ 29. Further, Dr. Rubin opines that a person of ordinary skill in the art would “hold a bachelor’s degree or the equivalent in computer science (or related academic fields) and three to four years of additional experience in the field of computer security, or equivalent work experience.” *Id.*

Patent Owner, through its expert, Dr. Michael Goodrich, offers a level of ordinary skill that is different from Petitioner’s. Ex. 2019 ¶ 25. In Particular, Dr. Goodrich opines that a person of ordinary skill in the art would have a “bachelor’s degree in computer science or related field, and

IPR2015-01974  
Patent 7,647,633 B2

either (1) two or more years of industry experience and/or (2) an advanced degree in computer science or related field.” *Id.*

In comparison, it appears that the minimum experience under Patent Owner’s proffered level of skill is one year less than Petitioner’s. Also, Patent Owner proffers an alternative to work experience, namely an advanced degree. There is no specific articulation regarding how the difference of one year’s experience or the proposed alternative of advanced degree in lieu of experience tangibly affects our obviousness inquiry. Further, there is no evidence in this record that the differences noted above impact in any meaningful way the level of expertise of a person of ordinary skill in the art. Indeed, we note that Dr. Goodrich’s opinions would not change if he had considered instead the level of ordinary skill in the art proffered by Dr. Rubin. *Id.* ¶ 28.

Accordingly, we determine that in this case no express definition of the level of ordinary skill in the art is necessary and that the level of ordinary skill in the art is reflected by the prior art of record. *See Okajima v. Bourdeau*, 261 F.3d 1350, 1355 (Fed. Cir. 2001); *In re GPAC Inc.*, 57 F.3d 1573, 1579 (Fed. Cir. 1995); *In re Oelrich*, 579 F.2d 86, 91 (CCPA 1978).

#### E. OBVIOUSNESS GROUND BASED ON SHIN

Petitioner asserts that claims 14 and 19 are unpatentable as obvious over Shin. Pet. 38–44. Specifically, with regards to Shin, Petitioner contends that Shin discloses Python or Java code (*id.* at 38–39) with software modules (*id.* at 39) implemented in an HTTP proxy server or “re-communicator” and a Java Virtual Machine within the HTTP client (*id.* at 40) to receive “messages” from the web server and to send those “messages”

IPR2015-01974  
Patent 7,647,633 B2

to the client (*id.* at 41). At the HTTP client, according to Petitioner, Shin executes safeguarding code incorporated in modified applets to process the operations attempted by the applet, “such as window attacks, network accesses, and uniform resource locator (URL) spoofing, by performing security checks and raising exceptions if those checks fail.” *Id.* at 42–43. Therefore, Shin, according to Petitioner, teaches all the limitations recited in claim 14. As for claim 19, Petitioner alleges that Shin teaches the further limitation of “at least one of a firewall and a network server” because Shin teaches the use of an HTTP proxy server that modifies classes before they are received by the browser. *Id.* at 44.

*1. Overview of Shin (Ex. 1009)*

Shin’s title is “Java Bytecode Modification and Applet Security.” Ex. 1009, 1. By its very title, Shin describes modifying bytecode. The reason for the modification is to protect client machines from applets that a Java Virtual Machine alone cannot protect. *Id.* at 3–4 (describing denial of service attacks, disclosure of confidential information attack, spoofing attack, and annoyance attack). Shin describes restricting the applet’s operations by “inserting safeguarding code,” that monitors and controls resource usage and that limits the functionality of the applet. *Id.* at 4. In practice, Shin substitutes one “executable entity, such as a class or a method, with a related executable entity that performs additional run time tests.” *Id.* The applet is modified at the HTTP proxy server, which sits between a web server and a client browser. *Id.*

Shin describes two ways to implement the modification: class-level modification and method-level modification. *Id.* at 4–5. Shin states that “[i]n Java, all references to strings, classes, fields, and methods are resolved

IPR2015-01974

Patent 7,647,633 B2

through indices into the constant pool of the class file, where their symbolic names are stored.” *Id.* at 5 (citation omitted). For a Java class-file change, Shin modifies the constant pool. For example, a “Window” class may be replaced by a sub-class, “Safe\$Window,” that sets a limit on the number of new windows that an applet may create. *Id.* That substitution is shown in Figure 2 of Shin, reproduced below.

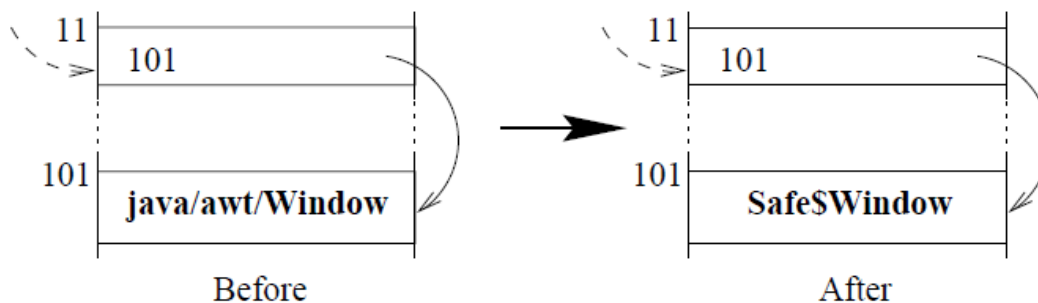


Figure 2: Class-level modification substitutes class reference

Figure 2 illustrates that the constant pool is altered to replace the entry 101, the name of the “class java/awt/Window,” with the name of the new class, “Safe\$Window.” *Id.* at 6.

As for the method-level modification, Shin also replaces a method with a related method. *Id.* at 6. As in the case of the class-level modification, Shin describes making substitutions in the constant pool. *Id.* at 7–8. The method reference entry is modified and new entries are added to reference the new method. *Id.* 8–9. Yet another technique that Shin discloses is the method invoking modification. *Id.* at 9. In this technique, the new method, such as “Safe\$Thread.setPriority(Ljava/lang/Thread;I)V” is added into the bytecode program. *Id.* at 9. In short, Shin’s class-level modification requires constant pool modifications, while method-level

IPR2015-01974  
Patent 7,647,633 B2

modifications requires constant pool modifications and instruction modifications. Regardless of which modification is desired, Shin explains that the techniques involve “modified executable entities [that] are inserted in Java bytecode.” *Id.* at 4.

## 2. Analysis

The issue for us to decide is whether Petitioner has shown by a preponderance of the evidence that Shin teaches or suggests the recited “executable code” as we have construed the term. Patent Owner argues that Shin teaches modification of executable code, because Shin describes modifying the Java applets at the HTTP proxy server with “safeguarding code” before forwarding the modified applet to the client. PO Resp. 27–28. In response, Petitioner argues that Shin’s applet modification is “essentially the same as the application programming interface (API) import address table (IAT) modification taught in the ’633 patent.” Reply 16 (citing Ex. 1001, 17:43–67). Specifically, Petitioner asserts that the Shin modification, like the IAT modification, involves “modifying pointers to resources used by the device executing the executable code.” *Id.* (citing Ex. 2022, Dr. Rubin’s cross-examination testimony, 138:16–139:1). During the oral argument, Petitioner clarified its contention that the constant pool in Shin is modified and the constant pool is “located in the Java Bytecodes of the Java applet.” Tr. 23:7–20. In sum, Petitioner contends that Shin’s constant pool does not constitute executable code, even though it is part of the downloadable delivered to the client. *Id.* at 23:21–24:24.

We find that Shin’s constant pool is not the same as the IAT embodiment described in the ’633 patent. As stated above, the mobile protection code modifies the IAT at the destination while processing the one

IPR2015-01974  
Patent 7,647,633 B2

or more operations of the executable code. In contrast, Shin’s applet is modified at the proxy server, and its modification constitutes rewriting the bytecode of the applet, which is executable. *See* Ex. 1009, 2 (“we propose a technique, called *bytecode modification*, through which we put restrictions on applets by inserting additional bytecode instructions that will perform the necessary run-time tests”), 4 (“modified executable entities are inserted in Java bytecode”). As explained by Shin, the constant pool is part of the class file of the applet. *Id.* at 5. Further, the bytecode modification is not performed at run-time, like the IAT modification table scenario. *See id.* at 2 (“HTTP proxy server []modifies classes before they are received by the browser”), 4 (Shin explaining that the “safety mechanism must be applied before the applet is executed”).

The differences are significant, for the modification of the applet at the server is precisely the type of modification that the ’633 patent sought to avoid. The applet is undisputedly executable code, and is what Petitioner pointed to in Shin as disclosing executable code. *See* Pet. 43 (“Shin also discloses that the safeguarding code processes operations attempted by the applet (‘executable code’) . . .”). By describing that the server modifies the constant pool in the applet, we find that Shin teaches modifying the executable code before the applet is processed at the client browser. Therefore, Petitioner has failed to show by a preponderance of the evidence that Shin teaches or suggests that the applet is “executable code” as we have construed the claim language, *supra*.

We are not persuaded by Petitioner’s arguments to the contrary. For instance, Petitioner argues that the modification of Shin’s applet does not change “operations of the original bytecode,” and, therefore, original

IPR2015-01974

Patent 7,647,633 B2

executable code is not altered. Reply 17 (citing Ex. 2022, Rubin cross-examination testimony, 26:6–17). First, we do not agree with Petitioner that the construction requiring no modification of executable code “only preclude[s] modifying the operations of the executable code[,] rather than [modification of] the code itself.” *Id.* Petitioner has this backward. The claim language, as we stated above, may encompass modification of the operations, as this is expressly described in the ’633 patent when the mobile protection code processes those operations at the destination. However, the modification of the operation during processing, allowed by the claim, is distinct from any modification of the code at a server. The ’633 patent, again, ensured the former, while avoiding the latter.

Second, we glean from Petitioner’s argument above that the applet Petitioner pointed to in the Petition is alleged to include both the “executable code” and the “mobile protection code.” Pet. 42 (“Shin also discloses a Java Virtual Machine (‘mobile code executor’) that can run applets that are modified to include safeguarding code . . .”). Petitioner, however, has the burden of proving that Shin’s applet constitutes both executable code and mobile protection code as claimed. Faced with the evidence of Shin’s constant pool modification, Petitioner argued that the constant pool portion of the applet file is not executable code. Reply 16–17 (“Shin’s applet modification . . . does not involve modifying problematic executable code—it only modifies the lookup table referenced when a method or class is called.”). Although we have found this argument unpersuasive, we note that even if we had not, Petitioner fails to explain whether Shin’s constant pool is mobile protection code or something else entirely. For example, at oral argument, Petitioner belatedly argued that the substitution with a class



IPR2015-01974

Patent 7,647,633 B2

Safe\$Window is “the only modification that the class-level modification makes in Shin[, and] that modification effectuates the mobile protection code’s protection.” Tr. 22:16–23. As discussed above, we understand Shin to implement its safeguarding code by rewriting the constant pool portion of the applet file. Ex. 1009, 4–5 (explaining that “modified executable entities are inserted in Java bytecode” and that “the modifications may be divided into two general forms,” one of which is the class-level modification that modifies the new class reference in the constant pool). Thus, if we were to identify Shin’s constant pool with the claimed mobile protection code, we find that such a mapping would fail. According to the claim language, the mobile protection code is executed, and according to Petitioner, the constant pool is only a lookup table (Reply 16–17). Further, the claim language requires that the mobile protection code process the attempted operations of the executable code, and Petitioner has not shown that the constant pool processes any operations.

According to our determination that the claimed executable code is not modified, and after review of the parties’ proffered evidence and arguments, we determine that Petitioner has not shown by a preponderance of the evidence that Shin teaches the claimed “executable code” and that Shin’s applet constitutes both “executable code” and “mobile protection code.”

### *3. Conclusion*

Petitioner has relied solely on Shin as teaching or suggesting all the limitations of claim 14. Claim 19 depends from claim 14. Consequently, based on our findings and conclusions stated above, we determine that

IPR2015-01974  
Patent 7,647,633 B2

Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable over Shin.

F. OBVIOUSNESS GROUND BASED ON POISON JAVA AND BROWN

With regard to Poison Java, Petitioner contends that AppletTrap is a system including code (Pet. 51) in distinct software modules (*id.*) where a web browser at the client receives instrumented applets and the HTML page (*id.* at 52). Petitioner also contends that Brown discloses running applets in a Java-enabled web browser. *Id.* In particular, Petitioner contends that Poison Java teaches wrapping monitoring code around the applet, such that the monitoring code can monitor and intercept potentially malicious code. *Id.* at 53–54. Petitioner argues that Poison Java “processes—and if appropriate, blocks—the executable code’s operations.” *Id.* at 54. Therefore, according to Petitioner, Poison Java teaches all the claim limitations of claim 14 with the added teaching of a Java-enabled web browser from Brown. As for claim 19, Petitioner also contends that Poison Java teaches the further limitation of “at least one of a firewall and a network server” because Poison Java describes a proxy server. *Id.*

1. *Overview of Poison Java (Ex. 1004)*

Poison Java describes “a hybrid solution to supplementing Java security” that integrates client- and sever-based solutions. Ex. 1004 at 42.<sup>9</sup> The solution is called InterScan AppletTrap. *Id.* AppletTrap pre-filters applets at the server, which performs more thorough examination of applets that include calls to system resources. *Id.* Poison Java describes the preparation process as instrumentation. *Id.* During this instrumentation,

---

<sup>9</sup> Citations to Poison Java refer to the pagination in the original article.

IPR2015-01974  
Patent 7,647,633 B2

“AppletTrap wraps monitoring code around the applet and attaches the security policy that determines what system resources it can access.” *Id.* Poison Java also describes instrumentation as “addition of extra code” that “deprives the applets of their original signatures.” *Id.*

Poison Java states that “[t]he HTML page, along with the instrumented applets, is then delivered to the client and displayed on its Web browser.” *Id.* at 43. AppletTrap’s monitoring code may permit the applet’s actions to take place, may block the action, and/or notify the user. *Id.* Poison Java describes AppletTrap with an advantage of “centralized protection, with control, deployment, and management functions at the server.” *Id.*

## 2. Analysis

With regard to Poison Java, the issue for us to decide is whether applets instrumented at the AppletTrap server constitute “executable code,” as claimed. We determine that they do not. Petitioner points to applets received at the proxy server as the executable code received at the re-communicator. Pet. 53. Petitioner then points to the monitoring code that is wrapped around the applet during instrumentation as mobile protection code. *Id.* at 54. Petitioner explains instrumentation as follows:

Code rewriting or instrumentation involves inserting extra protection code into the mobile code, e.g., at a server that receives the mobile code for transmission to a client computer. Instrumentation is performed after mobile code is filtered or scanned but before it is executed in the sandbox. Instrumentation often involves taking advantage of the fact that mobile code is usually divided into functions for better organization, code-reuse, and readability. Mobile code can be rewritten to provide an additional layer of

IPR2015-01974  
Patent 7,647,633 B2

security by inserting code right at the very beginning of a function body. Every time that function is called, the newly inserted code is executed first.

*Id.* at 9–10 (internal citations omitted).

Patent Owner contends that the instrumentation in Poison Java modifies the executable code. PO Resp. 36. We agree with Patent Owner’s contention and supporting evidence. In particular, Patent Owner points out that instrumentation is rewriting the applet, which is executable code. Petitioner’s description of instrumentation is consistent with Patent Owner’s characterization of Poison Java. As the Petition states, instrumentation adds extra protection code to the mobile code. Pet. 9. The addition of this code is performed through code rewriting. *Id.* (citing Ex. 1020, 2). Code rewriting, for a Java applet, involves transforming a binary program into a different but functionally equivalent program. Ex. 1020 at 2; *see also* Ex. 2019 ¶ 69. Having reviewed the facts presented and the arguments made on the record, we find that the instrumentation of the Java applet in Poison Java is a modification of the executable code, as the applet’s *binary code is rewritten at the server* to add functionality that monitors the applet’s behavior.

Our determination is further supported by our finding that the instrumentation described in Poison Java is the same type of modification that the ’633 patent described and distinguished from Shuang-Ji (Ex. 2006). As observed in our claim construction discussion, Shuang-Ji, in contrast with the invention described in the ’633 patent, modifies the downloadable component at the server. Shuang-Ji, as stated above, identifies problematic instructions, which “are then each instrumented, e.g., special code is inserted before and after each problematic instruction, where the special code calls

IPR2015-01974

Patent 7,647,633 B2

respectively a prefilter and a post filter.” Ex. 2006, 3:24–29. Poison Java, although more generic in its description of instrumentation, describes the applet instrumentation at the server, the same technique described in Shuang-Ji. Therefore, we determine that Petitioner has failed to demonstrate by a preponderance of the evidence that Poison Java discloses that “executable code” is not modified in accordance with our claim construction, *supra*.

Petitioner’s arguments to the contrary are not persuasive. In its Reply, Petitioner argues that Poison Java’s applet instrumentation is “essentially the same as the application programming interface (API) import address table (IAT) modification taught in the ’633 patent.” Reply 21 (internal citations omitted). In support of this statement, Petitioner argues that “[b]oth involve modifying pointers to resources used by the device executing the executable code.” *Id.* Petitioner cites expert testimony of Dr. Bims, one of Patent Owner’s experts. *Id.* (citing Ex. 1098, 64:19–68:18). None of the evidence proffered by Petitioner supports the contention that Poison Java involves pointers to resources. Furthermore, the cited testimony of Dr. Bims does not address Poison Java at all. Finally, Petitioner’s argument that Poison Java does not change operations at the destination is misplaced. Reply 21–22. As we discussed above with respect to Shin, the ’633 patent claims do not preclude modification of operations processed by the mobile operation code. Indeed, modification of the operations is the only modification discussed in the ’633 patent. The dispositive issue is whether the executable code at the destination is modified. And, as stated above, rewriting or instrumentation of the Java applet at the server is a modification of executable code.

IPR2015-01974  
Patent 7,647,633 B2

Therefore, Poison Java does not disclose, teach, or suggest the “executable code” phrase according to our construction, *supra*.

### 3. Conclusion

Petitioner has relied solely on Poison Java as disclosing the “executable code” phrase of claim 14. Brown is not alleged to cure the deficiencies noted above. Claim 19 depends from claim 14. Consequently, based on our findings and conclusions stated above, we determine that Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable over Poison Java and Brown.

### III. MOTIONS TO EXCLUDE

Petitioner moves to exclude various portions of the record as follows:

- 1) Paragraph 57 of Ex. 2019, Goodrich Declaration, on the basis of insufficient factual support and unreliability. Paper 35, 1–3.
- 2) Paragraphs 13–27 and 30–34 of Ex. 2020, Bims Declaration, on the basis of improper expert testimony. *Id.* at 4–8.
- 3) Exs. 2004, 2007, 2018, 2012, 2014, 2016, 2027, 2028, 2029, 2037, and 2048 in general as irrelevant, potentially confusing, and prejudicial. *Id.* at 8–15.

Petitioner’s Motion to Exclude is denied as moot, because that evidence objected to is not relied upon in reaching our determination that Petitioner has not met its burden of showing that claims 14 and 19 are unpatentable.

Patent Owner moves to exclude Exhibits 1002, 1004, 1005, 1006, 1007, 1008, 1009, 1041, 1082, 1092, 1093, 1095, 1099, 1101, and 1035. Paper 36 (“Mot.,” “PO Motion to Exclude”).

IPR2015-01974  
Patent 7,647,633 B2

*1. Exhibit 1002, Dr. Rubin's Declaration*

Patent Owner urges that the Board exclude paragraphs 79–83 of Dr. Rubin's Declaration regarding claim construction on the basis that Patent Owner alleges Petitioner did not permit Patent Owner to question Dr. Rubin regarding those opinions. Mot. 1–3. We deny Patent Owner's request as moot, because we do not rely on any testimony from Dr. Rubin's Declaration regarding claim construction.

*2. Exhibit 1005, Declaration of Mr. Grenier*

Patent Owner seeks exclusion of the Grenier Declaration, Exhibit 1005, on the basis of lack of authentication, relevance, and because the witness lacks personal knowledge. Mot. 3–4. Patent Owner alleges that Mr. Grenier's testimony has no relevance to the prior art at issue. *Id.* We do not agree. Mr. Grenier testifies regarding the accessibility, either in print or via database of the Poison Java article published in the IEEE Spectrum magazine. His testimony is highly relevant to Poison Java's status as prior art. Furthermore, we do not agree with Patent Owner that Mr. Grenier's testimony should be excluded for lack of personal knowledge. The portion of the deposition testimony Patent Owner relies on for this assertion does not support Patent Owner's argument. To the contrary, we find that Mr. Grenier has demonstrated, as Senior Director of Publishing Technologies at IEEE, that he has knowledge of the business practices of IEEE with regard to publishing and maintaining accurate records of its publications. We also agree with Petitioner that Mr. Grenier's testimony confirms that the printed publication of August 1999 of the magazine article is reflected in the database entries, which are performed according to business practices of

IPR2015-01974  
Patent 7,647,633 B2

IEEE. Paper 39, 2–3. The objections to Exhibit 1005 are, therefore, overruled. Patent Owner’s request to exclude this Exhibit is denied.

*3. Exhibit 1006, “Author’s Webpage”*

Patent Owner seeks exclusion of Exhibit 1006 because it is allegedly hearsay, irrelevant, and lacks authentication. Mot. 4–5. Specifically, Patent Owner argues the lack of “independent authentication” and that this exhibit cannot demonstrate that Shin was publicly accessible. *Id.* Patent Owner also argues that the date of publication is hearsay and is irrelevant to establish the date of Shin. *Id.* We disagree, but need not address each of the above superficial arguments because we do not rely on Exhibit 1006 to render our decision that Shin is a printed publication as stated above. Patent Owner’s request is denied as moot.

*4. Exhibit 1007, Filewatcher.*

Patent Owner seeks exclusion of Exhibit 1007 on the basis of lack of authentication, hearsay, and irrelevance. Mot. 5–6. For the same reasons as stated above with regard to Exhibit 1006, we deny this request as moot.

*5. Exhibit 1008, Kava Paper*

Patent Owner seeks exclusion of Exhibit 1008 for lack of authentication and relevance. *Id.* at 5. For the same reasons stated above, with regard to Exhibit 1006, we deny this request as moot.

*6. Exhibit 1082, Kent Declaration*

Patent Owner contends that Mr. Kent lacks personal knowledge to testify regarding publication of Brown. *Id.* at 6–7. We do not agree. Mr. Kent has personal knowledge of Brown’s status as a printed publication. Whether Mr. Kent’s testimony is credible based on his reliance on different versions of Brown is an issue of weight, not admissibility. Indeed, Patent



IPR2015-01974

Patent 7,647,633 B2

Owner cross-examined the witness, and through that cross-examination (Ex. 2024), we find that the declaration testimony was amended to reflect more accurately what Mr. Kent perceived as the earliest date of publication of Brown. Patent Owner's objections to Exhibit 1082 are overruled, and the request to exclude it is, therefore, denied.

*7. Exhibit 1093, Sherfesse Affidavit*

Patent Owner makes several arguments in support of its contention that Exhibit 1093 is inadmissible. *Id.* at 7–8. We do not address any of those arguments for we do not rely on Exhibit 1093 to render our decision. Patent Owner's request is, therefore, denied as moot.

*8. Exhibits 1092 and 1095, Christopher Butler Affidavits*

Patent Owner seeks exclusion of Exhibits 1092 and 1095, which are declarations of Mr. Christopher Butler, the Office Manager at the Internet Archive, regarding two archived websites proffered as evidence tending to show public access and publication of Brown and Shin, respectively. *Id.* at 8. The basis for the request is alleged lack of personal knowledge of Mr. Butler. *Id.* We do not agree with Patent Owner's contention. Mr. Butler declares that he has personal knowledge of the facts testified to in his declaration, and none of Patent Owner's arguments and evidence show that this is not the case. Exs. 1092 and 1095 ¶ 1. Furthermore, Patent Owner cross-examined Mr. Butler (Ex. 2025), and confirmed through that cross-examination that Mr. Butler's responsibilities at the Internet Archive include processing requests for authentication of records from the website archive.org (*id.* at 5:10–21). Mr. Butler also testified more particularly about archived webpages, the timestamps for the webpages accessible through the URLs, and the common location of the URL at the footer of a

IPR2015-01974  
Patent 7,647,633 B2

printout. *See, e.g., id.* at 21:1–13. Mr. Butler has established how archive.org maintains the archived web pages and how, in its ordinary course of business, the printouts of archived webpages provide URLs with timestamps in accordance with the information in the affidavit. We find the testimony sufficient to show that Mr. Butler has personal knowledge and that the affidavit supports Petitioner’s contentions regarding the publications of Brown and Shin. *See supra*, Section II.B. Therefore, Patent Owner’s objections are overruled. Patent Owner’s request to exclude Exhibits 1092 and 1095 is denied.

9. *Portions of Exhibits 1099, 1101, 1035, and 2022, and  
Petitioner’s Reply.*

Patent Owner requests exclusion of the above-identified exhibits and the Petitioner’s Reply on the basis that they introduce arguments and evidence outside the proper scope of a reply. Mot. 9–11. The request is denied. We have stated repeatedly that a motion to exclude is not a vehicle for arguing that Petitioner’s arguments and supporting evidence are outside the proper scope of a reply.<sup>10</sup> A motion to exclude evidence filed for the purpose of striking or excluding an opponent’s brief and/or evidence that a party believes goes beyond what is permitted under 37 CFR § 42.23 is improper. An allegation that evidence does not comply with 37 CFR

---

<sup>10</sup> *See Valeo v. Magna Elecs., Inc.*, Case IPR2014-00227, Paper 44 (PTAB Jan 14, 2015); *Carl Zeiss SMT GmbH v. Nikon Corp.*, Case IPR2013-00362, Paper 23 (PTAB June 5, 2014); *Ultratec, Inc. v. Sorenson Commc’ns, Inc.*, Case IPR2013-00288, Paper 38 at 2 (PTAB May 23, 2014); *Primera Tech., Inc. v. Automatic Mfg. Sys., Inc.*, Case IPR2013-00196, Paper 33 (PTAB Feb. 10, 2014); *ZTE Corp. v. Contentguard Holdings Inc.*, Case IPR2013-00133, Paper 42 (PTAB Jan. 21, 2014).

IPR2015-01974

Patent 7,647,633 B2

§ 42.23 is not a sufficient reason under the Federal Rules of Evidence for making an objection and requesting exclusion of such evidence.

Accordingly, these arguments are not considered, and the request to exclude portions of Exhibits 1099, 1101, 1035, 2022, and the Petitioner's Reply as outside the scope is denied.

*10. Exhibits 1009 (Shin), 1004 (Poison Java), and 1041 (Brown)*

Patent Owner argues that Shin, Poison Java, and 1041 should be excluded based on lack of authentication, relevance, and hearsay. Mot. 12–15. We deny this request. First, we fail to understand the relevance argument. Patent Owner argues that the date on the face of the documents is irrelevant. Patent Owner mistakes relevancy with how much weight the evidence should be credited with. If a reference, such as Poison Java contains dates and other indicia showing that it was published by a certain timeframe, the fact that those dates appear on the face of the reference is relevant, unless it is untrustworthy. Second, these three references (Shin, Poison Java, and Brown) have been shown to be prior art printed publications through various sources. Shin, for example, has been shown to be a printed publication though the evidence of the archived webpages showing that Shin was available publicly to anyone with Internet access and a search engine. Petitioner did not rely on Shin alone. For Poison Java and Brown, other evidence has been proffered to show their state as prior art printed publications. Again, Petitioner did not rely on dates on the face of the references alone. Notwithstanding the above analysis, we also note that Shin, Poison Java, and Brown are not hearsay, as they are being considered only for what they describe and not for truth. *See Fed. R. Evid. 807(c); Joy*

IPR2015-01974  
Patent 7,647,633 B2

*Techs., Inc. v. Manbeck*, 751 F.Supp. 225, 233 n.2 (D.D.C. 1990), *aff'd*, 959 F.2d 226 (Fed. Cir. 1992).

Finally, Patent Owner's argument that the references have not been authenticated is conclusory and lacks sufficiently persuasive reasoning. As the movant, Patent Owner has the burden of proof to establish that it is entitled to the requested relief. 37 C.F.R. § 42.20(c). This burden cannot be met by conclusory arguments and bare assertions of inadmissibility. The objections to Shin, Poison Java, and Brown are overruled. Patent Owner's request with regard to Exhibits 1009, 1004, and 1041 is denied.

#### IV. CONCLUSION

For the foregoing reasons, we conclude that Petitioner has failed to show by a preponderance of the evidence that claims 14 and 19 are unpatentable.

We also deny as moot the entirety of Petitioner's Motion to Exclude. Patent Owner's Motion to Exclude is denied as to Exhibits 1005, 1082, 1092, 1095, 1009, 1004, and 1041. The remainder of Patent Owner's Motion to Exclude is either denied as moot or not considered.

#### V. ORDER

In consideration of the foregoing, it is hereby:

ORDERED that claims 14 and 19 of the '633 patent have not been shown to be unpatentable;

FURTHER ORDERED that Petitioner's Motion to Exclude is denied as moot;

FURTHER ORDERED that Patent Owner's Motion to Exclude is denied as to Exhibits 1005, 1082, 1092, 1095, 1009, 1004, and 1041 is

IPR2015-01974

Patent 7,647,633 B2

denied, and the remainder of the Motion is denied as moot or not considered;  
and

FURTHER ORDERED that because this is a Final Written Decision, parties to the proceeding seeking judicial review of the decision must comply with the notice and service requirements of 37 C.F.R. § 90.2.

IPR2015-01974  
Patent 7,647,633 B2

PETITIONER:

Orion Armon  
Max Colice  
Jennifer Volk  
Brian Eutermoser  
COOLEY LLP  
[oarmon@cooley.com](mailto:oarmon@cooley.com)  
[mcolice@cooley.com](mailto:mcolice@cooley.com)  
[jvolkfortier@cooley.com](mailto:jvolkfortier@cooley.com)  
[beutermoser@cooley.com](mailto:beutermoser@cooley.com)

Michael Rosato  
Andrew Brown  
WILSON SONSINI GOODRICH & ROSATI  
[mrosato@wsgr.com](mailto:mrosato@wsgr.com)  
[asbrown@wsgr.com](mailto:asbrown@wsgr.com)

PATENT OWNER:

James Hannah  
Jeffrey H. Price  
KRAMER LEVIN NAFTALIS & FRANKEL LLP  
[jhannah@kramerlevin.com](mailto:jhannah@kramerlevin.com)  
[jprice@kramerlevin.com](mailto:jprice@kramerlevin.com)

Michael Kim  
FINJAN INC.  
[mkim@finjan.com](mailto:mkim@finjan.com)

# **EXHIBIT 7**



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE  
 United States Patent and Trademark Office  
 Address: COMMISSIONER FOR PATENTS  
 P.O. Box 1450  
 Alexandria, Virginia 22313-1450  
 www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
90/013,652	12/09/2015	7647633	FINREXM0009	8419

115222 7590 05/10/2016  
 Bey & Cotropia PLLC (Finjan Inc.)  
 Dawn-Marie Bey  
 213 Bayly Court  
 Richmond, VA 23229

EXAMINER
----------

BASEHOAR, ADAM L

ART UNIT	PAPER NUMBER
----------	--------------

3992

MAIL DATE	DELIVERY MODE
-----------	---------------

05/10/2016

PAPER

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.





UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents  
United States Patent and Trademark Office  
P.O. Box 1450  
Alexandria, VA 22313-1450  
[www.uspto.gov](http://www.uspto.gov)

**DO NOT USE IN PALM PRINTER**

(THIRD PARTY REQUESTER'S CORRESPONDENCE ADDRESS)

BRYAN CAVE LLP

1290 AVENUE OF THE AMERICAS

NEW YORK, NY 10104

**EX PARTE REEXAMINATION COMMUNICATION TRANSMITTAL FORM**

REEXAMINATION CONTROL NO. 90/013,652.

PATENT NO. 7647633.

ART UNIT 3992.

Enclosed is a copy of the latest communication from the United States Patent and Trademark Office in the above identified *ex parte* reexamination proceeding (37 CFR 1.550(f)).

Where this copy is supplied after the reply by requester, 37 CFR 1.535, or the time for filing a reply has passed, no submission on behalf of the *ex parte* reexamination requester will be acknowledged or considered (37 CFR 1.550(g)).

<b>Notice of Intent to Issue Ex Parte Reexamination Certificate</b>	<b>Control No.</b> 90/013,652	<b>Patent Under Reexamination</b> 7647633	
	<b>Examiner</b> ADAM L. BASEHOAR	<b>Art Unit</b> 3992	<b>AIA (First Inventor to File) Status</b> No

**-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --**

1.  Prosecution on the merits is (or remains) closed in this *ex parte* reexamination proceeding. This proceeding is subject to reopening at the initiative of the Office or upon petition. Cf. 37 CFR 1.313(a). A Certificate will be issued in view of

(a)  Patent owner's communication(s) filed: \_\_\_\_\_.

(b)  Patent owner's failure to file an appropriate timely response to the Office action mailed: \_\_\_\_\_.

(c)  Patent owner's failure to timely file an Appeal Brief (37 CFR 41.31).

(d)  The decision on appeal by the  Board of Patent Appeals and Interferences  Court dated \_\_\_\_\_

(e)  Other: The Order Granting Ex Parte Reexamination mailed 02/03/2016.

2. The Reexamination Certificate will indicate the following:

(a) Change in the Specification:  Yes  No

(b) Change in the Drawing(s):  Yes  No

(c) Status of the Claim(s):

(1) Patent claim(s) confirmed: 8 and 12.

(2) Patent claim(s) amended (including dependent on amended claim(s)): \_\_\_\_\_

(3) Patent claim(s) canceled: \_\_\_\_\_.

(4) Newly presented claim(s) patentable: \_\_\_\_\_.

(5) Newly presented canceled claims: \_\_\_\_\_.

(6) Patent claim(s)  previously  currently disclaimed: \_\_\_\_\_

(7) Patent claim(s) not subject to reexamination: 1-7,9-11 and 13-41.

3.  A declaration(s)/affidavit(s) under **37 CFR 1.130(b)** was/were filed on \_\_\_\_\_.

4.  Note the attached statement of reasons for patentability and/or confirmation. Any comments considered necessary by patent owner regarding reasons for patentability and/or confirmation must be submitted promptly to avoid processing delays. Such submission(s) should be labeled: "Comments On Statement of Reasons for Patentability and/or Confirmation."

5.  Note attached NOTICE OF REFERENCES CITED (PTO-892).

6.  Note attached LIST OF REFERENCES CITED (PTO/SB/08 or PTO/SB/08 substitute).

7.  The drawing correction request filed on \_\_\_\_\_ is:  approved  disapproved.

8.  Acknowledgment is made of the priority claim under 35 U.S.C. § 119(a)-(d) or (f).

a)  All b)  Some\* c)  None of the certified copies have

been received.

not been received.

been filed in Application No. \_\_\_\_\_.

been filed in reexamination Control No. \_\_\_\_\_.

been received by the International Bureau in PCT Application No. \_\_\_\_\_.

\* Certified copies not received: \_\_\_\_\_.

9.  Note attached Examiner's Amendment.

10.  Note attached Interview Summary (PTO-474).

11.  Other: \_\_\_\_\_.

**All correspondence** relating to this reexamination proceeding should be directed to the **Central Reexamination Unit** at the mail, FAX, or hand-carry addresses given at the end of this Office action.

	/ADAM L BASEHOAR/ Primary Examiner, Art Unit 3992
--	--

cc: Requester (if third party requester)

Application/Control Number: 90/013,652  
Art Unit: 3992

Page 2

### **DETAILED ACTION**

1. This Office Action addresses original claims 8 and 12 of United States Patent Number 7,647,633 B2 (Edery et al.), for which it has been determined in the Order Granting *Ex Parte* Reexamination (hereafter the “Order”) mailed 02/03/2016 that a substantial new question of patentability was raised in the Request for *Ex Parte* reexamination filed on 12/09/2015 (hereafter the “Request”). This Office Action is a Notice of Intent to Issue *Ex Parte* Reexamination Certificate (“NIRC”) and is in response to the filed Request and the subsequent Order. Claims 8 and 12 are allowable and/or confirmed as discussed below. Claims 1-7, 9-11, and 13-41 were not subject to this reexamination.

2. The Notice of Proceedings Pursuant to 37 C.F.R. 1.565(a), filed by Patent Owner (PO) on 04/04/2016, has been entered into the reexamination file and has been considered by the Examiner (see: MPEP 2282).

### **Information Disclosure Statement**

3. Regarding Information Disclosure Statement (IDS) submissions, MPEP 2256 recites the following: “Where patents, publications, and other such items of information are submitted by a party (patent owner or requester) in compliance with the requirements of the rules, the requisite degree of consideration to be given to such information will be normally limited by the degree to which the party filing the information citation has explained the content and relevance of the information. The initials of the examiner placed adjacent to the citations on the form PTO/SB/08A and 08B or its equivalent, without an indication to the contrary in the record, do

Application/Control Number: 90/013,652

Page 3

Art Unit: 3992

not signify that the information has been considered by the examiner any further than to the extent noted above.”

Accordingly, the IDS submission filed by Patent Owner on 03/10/2016 has been considered by the Examiner only with the scope required by MPEP 2256, unless otherwise noted. Additionally, the Supplemental IDS submission (related to the IDS submission noted above) filed by Patent Owner on 04/12/2016 has been considered by the Examiner only with the scope required by MPEP 2256, unless otherwise noted.

#### **References Discussed in Action**

4. The three prior art references discussed in this Notice of Intent to Issue *Ex Parte* Reexamination Certificate (“NIRC”), are as follows:

- **Ji** – (U.S. Patent No. 5,983,348, filed 09/10/1997, published 11/09/1999)
- **Shin** – (“Java Bytecode Modification and Applet Security”, Stanford University, Computer Science Department, Stanford CA (1998), pp. 1-20)
- **Martin** – (“Blocking Java Applets at the Firewall”, IEEE Computer Society Proceedings of the 1997 Symposium on Network and Distributed Systems Security Washington, DC (1997), pp. 16-26)

#### **Related Proceedings**

5. On 03/29/2016 a decision (hereafter the “PTAB 2015-01974 Decision”) was rendered (“Partial Institution of *Inter Partes* Review”) by the Patent Trial and Appeal Board (PTAB) in IPR2015-01974. The PTAB 2015-01974 Decision ordered that the Petition was denied with

Application/Control Number: 90/013,652

Page 4

Art Unit: 3992

regard to all grounds asserted for claims 1-4, 6-8, 13, 28, and 34 of the Edery '633 patent (PTAB 2015-01974 Decision: pp. 12-13, and 16). The PTAB 2015-01974 Decision further ordered that the Petition was granted for claims 14 and 19 of the Edery '633 patent (PTAB 2015-01974 Decision: pp. 13-16). More specifically, the PTAB 2015-01974 Decision determined that the technology for which the Shin reference is relied upon in the Petition is substantially the same as that which was considered relevant in the Ji reference (PTAB 2015-01974 Decision: pp. 9-10). Additionally, the PTAB 2015-01974 Decision notes that the Board has rendered a decision in related reexam 90/013,017 involving U.S. Patent No. 7,058,822, which is the parent of the Edery '633 patent and includes claims with claim terms similar to those recited in the Edery '633 patent (PTAB 2015-01974 Decision: p. 11).

In said related reexam 90/013,017, a decision (hereafter the "PTAB 90/013,017 Decision") on appeal by the Patent Trial and Appeal Board (PTAB) was rendered on 12/30/2015 and mailed reversing the Examiner's decision rejecting claims 1-8, 16-27, 37, and 40 of the Edery '822 patent (PTAB 90/013,017 Decision: p. 15). More specifically, the PTAB 90/013,017 Decision determined, regarding substantially similar claim limitations, that in order to disclose determining *whether* the downloadable-information includes executable code, a reference must disclose distinguishing between two alternative possibilities: executable code is included in the downloadable-information, and executable code is not included in the downloadable-information (PTAB 90/013,017 Decision: pp. 5, 9, and 12-13). Further, it was determined (PTAB 90/013,017 Decision: p. 5: "Therefore, Ji does not disclose the recited determining") that while the Ji reference does determine some cases where executable code is included in the downloadable-information, Ji does not adequately determine when executable code is not

Application/Control Number: 90/013,652

Page 5

Art Unit: 3992

included in the downloadable information (i.e., the lack of an applet tag does not determine that executable code is not included in the downloadable-information). Please also note the related information regarding the Edery '822 patent and the Shin reference determined in the decision (hereafter the "PTAB 2015-01999 Decision") rendered ("Denying Institution of *Inter Partes* Review") by the PTAB on 03/29/2016 (PTAB 2015-01999 Decision: pp. 8-10).

Finally, on 10/20/2014 the District Court (*Finjan, Inc. v. Blue Coat Systems, Inc.* - Case No. 5:13-cv-03999-BLF) issued a claim construction order (hereafter the "03999 Construction") pertaining to certain claims of the Edery '633 patent. The District Court construed, for at least independent claim 13 of the Edery '633 patent (03999 Construction: pp. 8 and 24), that the limitation "means for causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code" means "if the downloadable-information is determined to include executable code, causing mobile protection code to be communicated to at least one information-destination of the downloadable-information without modifying the executable code" (emphasis added). The District Court determined that it was clear from the specification of the Edery '633 patent that the mobile protection code (MPC) does not modify executable code found in the downloadable-information (03999 Construction: pp. 9-10).

#### **STATEMENT OF REASONS FOR PATENTABILITY AND/OR CONFIRMATION**

6. The following is an Examiner's statement of reasons for patentability and/or confirmation for claims 8 and 12. Claims 8 and 12 each being confirmed over the prior art that was explained in the Request and determined to raise a substantial new question of patentability in the Order

Application/Control Number: 90/013,652

Page 6

Art Unit: 3992

granting reexamination as discussed by the Examiner in the present reexamination proceeding, because of the following: Regarding independent claim 8, the proposed prior art does not explicitly teach or suggest the limitations of (1) “determining...whether the downloadable-information includes executable code” and (2) “for causing mobile protection code (“MPC”) to be communicated...if the downloadable-information is determined to include executable code”, in combination with the remaining elements or features of the claimed invention.

A. The Ji reference has been predominantly cited in the Request to teach said limitations in independent claim 8 (see: Request, pp. 23-27). However, the Ji reference appears to only teach both static and dynamic scanning for application programs (e.g., Java applets or ActiveX controls) received at a conventional (proxy) server (see: Ji, column 3, lines 16-25). Ji teaches that upon receipt of a particular Java applet, the applet is scanned and instrumented. Ji further teaches wherein downloaded non-applets are not scanned (see: Ji, column 4, line 66-column 5, line 4). As noted above in the Related Proceedings section, the PTAB determined that in order to disclose determining *whether* the downloadable-information includes executable code, a reference must disclose distinguishing between two alternative possibilities: executable code is included in the downloadable-information, and executable code is not included in the downloadable-information. Therefore, while Ji does determine some instances where executable code (e.g., an applet tag or particular instructions) is included in the downloadable-information, Ji does not teach or suggest the recited “determining” limitation because Ji does not adequately determine when executable code is not included (e.g., see: Edery ‘633, column 12, lines 20-21:

Application/Control Number: 90/013,652

Page 7

Art Unit: 3992

“a determined non-executable (“NXEQ”)) in the downloadable-information (PTAB 90/013,017

Decision: p. 5: “Therefore, Ji does not disclose the recited determining”).

The Ji reference also explicitly teaches that when the mobile protection code is to be communicated to at least one information-destination of the downloadable-information, an instrumenter of the conventional (proxy) server modifies the applet (i.e., the executable code) of the downloadable-information (see: Ji, column 3, lines 25-35: “identified problematic instructions are then each instrumented, e.g. special code is inserted before and after each problematic instruction...replacing the problematic instruction with another instruction...instrumented applet is then downloaded”; column 5, lines 15-27: “The present applet scanner thus uses applet instrumentation technology...it alters the Java applet byte code sequence during downloading of the applet to the server”; column 6, line 38-column 7 line 1: “the content of the applets is changed by the instrumentation”). As noted above in the Related Proceedings section, the District Court determined that it was clear from the specification of the Edery ‘633 patent that the mobile protection code (MPC) does not modify executable code found in the downloadable-information (03999 Construction: pp. 9-10). The Examiner agrees with the claim construction as the limitations of construed independent claim 13 are substantially similar to those of currently examined independent claim 8. For example, the Edery ‘633 patent discloses advantageously protecting against unknown mobile code without modifying the mobile code (see: Edery ‘633, column 4, lines 12-16: “Advantageously...without modifying the mobile code”; column 10, lines 39-45: “apparent that performing executable code detection and communicating to a downloadable destination an MPC...as separate from a detected-Downloadable is more accurate and far less resource intensive than...modifying a



Application/Control Number: 90/013,652

Page 8

Art Unit: 3992

Downloadable”). The Edery ‘633 patent is silent on any clear examples wherein the executable code is modified (i.e., at the server/firewall). Therefore, while Ji does teach for causing mobile protection code (“MPC”) to be communicated to at least one information-destination of the downloadable-information, Ji does not teach or suggest doing so without directly modifying the executable code of the downloadable-information.

B. The Shin reference has also been predominantly cited in the Request to teach said limitations in independent claim 8 (see: Request, pp. 36-40). However, the Shin reference similarly fails to teach or suggest said limitations for the same reasons as discussed above with regard to the Ji reference. As noted above in the Related Proceedings section, the PTAB determined that the technology relied upon in the Shin reference is substantially the same as that which was considered relevant in the Ji reference (PTAB 2015-01974 Decision: pp. 9-10).

The Shin reference, via referencing the Martin reference discussed below, discusses techniques for identifying/detecting Java applets at an HTTP proxy server or firewall. Shin discloses that one idea is to look for <applet> tags in the download stream and another idea is to detect Java class files at the firewall by a magic byte sequence that is required at the beginning of every class file or by their name which will end in .class (see: Shin, p. 17-18). Therefore, similar to the Ji reference, Shin fails to teach or suggest the recited “determining” limitation because Shin does not adequately determine when executable code is not included in the downloadable-information. Likewise, the Shin reference does teach for causing mobile protection code (“MPC”) to be communicated to at least one information-destination of the downloadable-information (see: Shin, p. 1: “insert additional run-time tests into Java applets”; p. 2: “bytecode

Application/Control Number: 90/013,652

Page 9

Art Unit: 3992

modification...class-level modification and method-level modification...an HTTP proxy server that modifies classes before they are received by the browser”; p. 4: “substitutes one executable entity...with a related executable entity...applets are currently modified within an HTTP proxy”; pp. 5-7). However, similar to the Ji reference, Shin also does not teach or suggest doing so without directly modifying the executable code of the downloadable-information (see: Shin, p. 4: “modified executable entities are inserted in Java bytecode”).

C. The Martin reference has been predominantly cited in the Request in combination with the references discussed above to read on independent claim 8 (i.e., Shin in view of Martin) and read on dependent claim 12 (i.e., Ji in view of Martin or Shin in view of Martin). The Martin reference being relied upon to more specifically teach three strategies for detecting external Java applets (i.e., executable code) at a site’s firewall (see: Request, pp. 30-32 and 36-38). Martin further teaching that in order to detect said external Java applet, a proxy/firewall could: (1) “Rewriting <applet> Tags” (see: Martin, pp. 22-23); (2) “Blocking CA FE BA BE” (see: Martin, p. 23); and (3) “Blocking by Requested Filename” (see: Martin, pp. 23-24). However, despite providing more explicit teachings on how to detect Java applets at a proxy/firewall, the Martin reference does not cure the deficiencies found in the Ji and Shin references. By primarily searching for Java applets, and perhaps other popular portable-executable formats such as ActiveX (see: Martin, p. 23), the Martin reference does not appear to determine *whether* the downloadable-information includes executable code as discussed in detail above (i.e., the lack of an applet tag, the lack of the 4-byte hex signature CA FE BA BE, or the lack of a file name ending in .class does not determine that executable code is not included in the downloadable-

Application/Control Number: 90/013,652

Page 10

Art Unit: 3992

information). Additionally, because the intended purpose of the Martin reference was to block all external Java applets at the proxy/firewall (see: Martin, pp. 16-17: “various techniques to block Java applets at a site’s firewall so that internal applications can run Java in their browsers, but untrusted applets from the outside cannot penetrate”; p. 21: “blocking applets from crossing the firewall...Java attacks can be prevented”; p. 25: “There is no doubt that blocking applets from crossing a firewall is of essence”), Martin clearly does not teach or suggest causing mobile protection code (“MPC”) to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code. Similarly, it is not clear that one of ordinary skill in the art at the time of the invention would have found it obvious to combine said teachings of Martin with either Ji or Shin, because the Ji and Shin references each appear to teach away (allow all Java applets, albeit modified, to be forwarded to a requesting client side destination) from the principle of operation of Martin (block all external Java applets from being forwarded to a requesting client side destination).

Regarding dependent claim 12, the claim depends on confirmed independent claim 8 and is therefore also confirmed at least via dependency.

7. Any comments considered necessary by PATENT OWNER regarding the above statement must be submitted promptly to avoid processing delays. Such submission by the patent owner should be labeled: “Comments on Statement of Reasons for Patentability and/or Confirmation” and will be placed in the reexamination file.

Application/Control Number: 90/013,652

Page 11

Art Unit: 3992

### Conclusion

8. All correspondence relating to this *ex parte* reexamination proceeding should be directed as follows:

By U.S. Postal Service Mail to:

Mail Stop Ex Parte Reexam  
ATTN: Central Reexamination Unit  
Commissioner for Patents  
P.O. Box 1450  
Alexandria, VA 22313-1450

By FAX to:

(571) 273-9900  
Central Reexamination Unit

By hand to:

Customer Service Window  
Randolph Building  
401 Dulany St.  
Alexandria, VA 22314

By EFS-Web:

Registered users of EFS-Web may alternatively submit such correspondence via the electronic filing system EFS-Web, at

<https://efs.uspto.gov/efile/myportal/efs-registered>

EFS-Web offers the benefit of quick submission to the particular area of the Office that needs to act on the correspondence. Also, EFS-Web submissions are “soft scanned” (i.e., electronically uploaded) directly into the official file for the reexamination proceeding, which offers parties the opportunity to review the content of their submissions after the “soft scanning” process is complete.

Application/Control Number: 90/013,652

Page 12

Art Unit: 3992

Any inquiry concerning this communication or earlier communications from the Reexamination Legal Advisor or Examiner, or as to the status of this proceeding, should be directed to the Central Reexamination Unit at telephone number (571) 272-7705.

/Adam L Basehoar/

Primary Examiner, Art Unit 3992

Conferees:

/DENNIS BONSHOCK/

Primary Examiner, Art Unit 3992

/ALEXANDER KOSOWSKI/

Supervisory Patent Examiner, Art Unit 3992

# **EXHIBIT 8**

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL AND APPEAL BOARD

---

Palo Alto Networks, Inc.  
Petitioner

v.

Finjan, Inc.  
Patent Owner

U.S. Patent No. 7,647,633

Filing Date: June 22, 2005

Issue Date: Jan. 12, 2010

Title: Malicious Mobile Code Runtime Monitoring System and Methods

**DECLARATION OF AVIEL D. RUBIN IN SUPPORT OF PETITION FOR  
*INTER PARTES* REVIEW OF U.S. PATENT NO. 7,647,633**

*Inter Partes* Review No. 2015-01974

Declaration of Aviel D. Rubin  
 Petition for *Inter Partes* Review of Patent No. 7,647,633

**Table of Contents**

	<b>Page</b>
I. INTRODUCTION AND QUALIFICATIONS .....	1
A. Engagement Overview .....	1
B. Summary of Opinions .....	1
C. Qualifications and Experience .....	3
1. Education .....	3
2. Career .....	3
3. Publications:.....	7
4. Curriculum Vitae.....	8
D. Materials Considered.....	8
II. LEGAL PRINCIPLES USED IN THE ANALYSIS .....	13
A. Person Having Ordinary Skill in the Art (“POSA”) .....	13
B. Prior Art.....	15
C. Broadest Reasonable Interpretations.....	15
D. Standards for Anticipation and Obviousness .....	16
III. TECHNOLOGY TUTORIAL .....	28
IV. THE ’633 PATENT .....	41
A. Overview of the ’633 Patent.....	41
B. The Claims of the ’633 Patent.....	42
C. Interpretation of Claim Limitations in the ’633 Patent .....	44
1. Non-Means-Plus-Function Limitations .....	44
2. Means-Plus-Function Limitations.....	47
D. The Priority Claims of the ’633 Patent .....	50
V. OVERVIEW OF THE PRIOR ART .....	53
A. Overview of Poison Java.....	53
B. Overview of Shin.....	55
C. Overview of Brown.....	55
D. Poison Java, Shin, and Brown Are All Analogous Art.....	56



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

**Table of Contents**  
**Continued**

	<b>Page</b>
VI. ANALYSIS.....	57
A. Shin Renders Claims 1–4, 6–8, 13, 14, and 19 Obvious under 35 U.S.C. § 103(a).....	57
1. Independent Claim 1 .....	57
a. Claim element 1[b]: “receiving” limitation.....	57
b. Claim element 1[c]: “determining” limitation.....	57
2. Claim 3 .....	59
3. Claims 6 and 7.....	60
4. Claim 8 .....	61
5. Claim 13 .....	62
6. Independent Claim 14 .....	63
a. Claim element 14[a]: “computer program product”.....	63
b. Claim element 14[c]: “information re- communicator” and “mobile code executor” .....	63
c. Claim element 14[d]: “receiving” limitation.....	64
d. Claim element 14[e]: “causing” limitation.....	64
B. Poison Java in view of Shin Renders Claim 1 Obvious under 35 U.S.C. § 103(a).....	65
1. Independent Claim 1 .....	65
a. Claim element 1[c]: “determining” limitation.....	65
C. Poison Java in view of Brown Renders Claims 14, 19, and 34 Obvious under 35 U.S.C. § 103(a).....	67
1. Independent Claim 14 .....	67
a. Claim element 14[a]: “computer program product”.....	67
b. Claim element 14[c]: “information re- communicator” and “mobile code executor” .....	68
c. Claim element 14[d]: “receiving” limitation.....	69

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

**Table of Contents  
Continued**

	<b>Page</b>
d. Claim element 14[e]: “causing” limitation.....	69
2. Independent Claim 34 .....	70
a. Claim element 34[b]: “mobile code executor” .....	70
VII. SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS .....	71
VIII. CONCLUSION.....	72

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

1. I, Aviel Rubin, declare as follows:

2. I have personal knowledge of the facts stated in this declaration, and could and would testify to these facts under oath if called upon to do so.

## **I. INTRODUCTION AND QUALIFICATIONS**

### **A. Engagement Overview**

3. I have been retained by counsel for Palo Alto Networks, Inc. in this case as an expert in the relevant art. I am being compensated for my work at the rate of \$688 per hour. No part of my compensation is contingent upon the outcome of this petition.

4. I was asked to study U.S. Patent 7,647,633 (the “’633 patent”), its prosecution history, and the prior art and to render opinions on the obviousness or non-obviousness of certain claims of the ’633 patent (the “Petitioned Claims”) in light of the teachings of the prior art, as understood by a person of ordinary skill in the art in the 2000-2001 timeframe.

### **B. Summary of Opinions**

5. After studying the ’633 patent, its file history, and the prior art, and considering the subject matter of the claims of the ’633 patent in light of the state of technical advancement in the field of mobile-code security in the 2000-2001 timeframe, I reached the following conclusions:

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

- a) By the year 2000, techniques for identifying mobile code in network traffic were already well known to those of ordinary skill in the art.
- b) By the year 2000, hybrid solutions for mobile-code security combining server/gateway-based scanning of data traffic and client-side runtime monitoring of mobile code instrumented at the server/gateway was already well known.
- c) By the year 2000, it was already well known to apply and enforce security policies on client computers in conjunction with runtime monitoring of the instrumented mobile code.
- d) By the year 2000, the concept of sandboxing mobile code through code rewriting or instrumentation was also well known.

6. I have reviewed Petitioner's Petition for *Inter Partes* Review of claims 1–4, 6–8, 13, 14, 19, 28, and 34 of the '633 patent (the "Petitioned Claims"), and I agree with all of the grounds of invalidity presented therein. In light of that review and my general conclusions above, and as explained in more detail throughout this declaration, it is therefore my opinion that each of the Petitioned Claims was invalid as obvious in the 2000-2001 timeframe in light of the knowledge of skill in the art at that time and the teachings, suggestions, and

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

motivations present in the prior art.

**C. Qualifications and Experience**

7. I possess the knowledge, skills, experience, training and the education to form an expert opinion and testimony in this matter. I have 22 years of experience in the field of computer science, and specifically, in Internet and computer security.

**1. Education**

8. I received my Ph.D. in Computer Science and Engineering from the University of Michigan, Ann Arbor in 1994, with a specialty in computer security and cryptographic protocols. My thesis was titled “Nonmonotonic Cryptographic Protocols” and concerned authentication in long-running networking operations.

**2. Career**

9. I will discuss my current position as a professor first, followed by a synopsis of my career and work from when I received my Ph.D. to the present.

10. I am currently employed as Professor of Computer Science at Johns Hopkins University, where I perform research, teach graduate courses in computer science and related subjects, and supervise the research of Ph.D. candidates and other students. Courses I have taught include Security and Privacy in Computing and Advanced Topics in Computer Security. I am also the Technical Director of the Johns Hopkins University Information Security Institute, the University’s focal

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

point for research and education in information security, assurance, and privacy.

The University, through the Information Security Institute's leadership, has been designated as a Center of Academic Excellence in Information Assurance by the National Security Agency and leading experts in the field. The focus of my work over my career has been computer security, and my current research concentrates on systems and networking security, with special attention to software and network security.

11. After receiving my Ph.D., I began working at Bellcore in its Cryptography and Network Security Research Group from 1994 to 1996. During this period I focused my work on Internet and Computer Security. While at Bellcore, I published an article titled "Blocking Java Applets at the Firewall" about the security challenges of dealing with JAVA applets and firewalls, and a system that we built to overcome those challenges.

12. In 1997, I moved to AT&T Labs, Secure Systems Research Department, where I continued to focus on Internet and computer security. From 1995 through 1999, in addition to my work in industry, I served as Adjunct Professor at New York University, where I taught undergraduate classes on computer, network and Internet security issues.

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

13. I stayed in my position at AT&T until 2003, when I left to accept a full time academic position at Johns Hopkins University. The University promoted me to full professor with tenure in April, 2004.

14. I serve, or have served, on a number of technical and editorial advisory boards. For example, I served on the Editorial and Advisory Board for the International Journal of Information and Computer Security. I also served on the Editorial Board for the Journal of Privacy Technology. I have been Associate Editor of IEEE Security and Privacy Magazine, and served as Associate Editor of ACM Transactions on Internet Technology. I am currently an Associate Editor of the journal Communications of the ACM. I was an Advisory Board Member of Springer's Information Security and Cryptography Book Series. I have served in the past as a member of the DARPA Information Science and Technology Study Group, a member of the Government Infosec Science and Technology Study Group of Malicious Code, a member of the AT&T Intellectual Property Review Team, Associate Editor of Electronic Commerce Research Journal, Co-editor of the Electronic Newsletter of the IEEE Technical Committee on Security and Privacy, a member of the board of directors of the USENIX Association, the leading academic computing systems society, and a member of the editorial board of the Bellcore Security Update Newsletter.

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

15. I have spoken on information security and electronic privacy issues at more than 50 seminars and symposia. For example, I presented keynote addresses on the topics “Security of Electronic Voting” at Computer Security 2004 Mexico in Mexico City in May 2004; “Electronic Voting” to the Secure Trusted Systems Consortium 5th Annual Symposium in Washington DC in December 2003; “Security Problems on the Web” to the AT&T EUA Customer conference in March, 2000; and “Security on the Internet” to the AT&T Security Workshop in June 1997. I also presented a talk about hacking devices at the TEDx conference in October, 2011.

16. I was founder and President of Independent Security Evaluators (ISE), a computer security consulting firm, from 2005-2011. In that capacity, I guided ISE through the qualification as an independent testing lab for Consumer Union, which produces Consumer Reports magazine. As an independent testing lab for Consumer Union, I managed an annual project where we tested all of the popular anti-virus products. Our results were published in Consumer Reports each year for three consecutive years.

17. I am currently the founder and managing partner of Harbor Labs, a software and networking consulting firm.

18. As is apparent from the above description, virtually my entire professional career has been dedicated to issues relating to information and



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

network security. Moreover, through my consulting work and my work at AT&T and Bellcore, I am familiar with the practical aspects of designing, analyzing, and deploying security applications in network environments.

### 3. Publications:

19. I am a named inventor on ten United States patents, all in the information security area. The patent numbers and titles as well as my co-inventors are listed on the attached curriculum vitae. (*See* Ex. 1084.)

20. In March, 2004, I was asked by the Federal Trade Commission to submit a report commenting on the viability and usefulness of a national do not e-mail registry. I submitted my report entitled “A Report to the Federal Trade Commission on Responses to Their Request For Information on Establishing a National Do Not E-mail Registry” on May 10, 2004.

21. I have also testified before Congress regarding the security issues with electronic voting machines and in the United States Senate on the issue of censorship. I also testified in Congress on November 19, 2013 about security issues related to the government’s Healthcare.gov web site.

22. I am author or co-author of five books regarding information security issues: *Brave New Ballot*, Random House, 2006; *Firewalls and Internet Security* (second edition), Addison Wesley, 2003; *White-Hat Security Arsenal*, Addison Wesley, 2001; *Peer-to-Peer*, O’Reilly, 2001; and *Web Security Sourcebook*, John

Declaration of Aviel D. Rubin  
 Petition for *Inter Partes* Review of Patent No. 7,647,633

Wiley & Sons, 1997. I am also the author of numerous journal and conference publications.

#### 4. Curriculum Vitae

23. Additional details of my education and employment history, recent professional service, patents, publications, and other testimony are set forth in my current curriculum vitae, attached to this declaration as Ex. 1084.

#### D. Materials Considered

24. My analysis is based on my experience in the computer industry since 1994, including the documents I have read and authored and systems I have developed and used since then.

25. Furthermore, I have reviewed the various relevant publications from the art at the time of the alleged invention and the invalidity proofs that are included in the Petition for *Inter Partes* Review of the '633 patent, to which this Declaration relates. Based on my experience as a person having ordinary skill in the art ("POSA") at the time of the alleged invention, the references accurately characterize the state of the art at the relevant time. Specifically, I have reviewed the following:

Exhibit No.	Description of Document
1001	U.S. Patent No. 7,647,633 ("Edery <i>et al.</i> ")
1003	90/013,016, Final Office Action ("633 Reexam") (May 22, 2015)

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

<b>Exhibit No.</b>	<b>Description of Document</b>
<b>1004</b>	Eva Chen “Poison Java” IEEE Spectrum (1999)
<b>1005</b>	2015-09-10 Declaration of Gerard P. Grenier in support of the “Poison Java” reference
<b>1006</b>	Webpage: Workshop and Miscellaneous Publications, Insik Shin
<b>1007</b>	Webpage: Filewatcher – 7/27/98
<b>1008</b>	Ian Welch and Robert Stroud “Kava – A Reflective Java Based on Bytecode Rewriting” (January 1999)
<b>1009</b>	Shin Insik and John C. Mitchell “Java Bytecode Modification and Applet Security” (1998)
<b>1010</b>	Carey Nachenberg “The Evolving Virus Threat”
<b>1011</b>	David M. Chess “Security Issues in Mobile Code Systems” (1998)
<b>1012</b>	R. Braden and J. Postel “Requirements for Internet Gateways” (June 1987)
<b>1013</b>	International Publication No. WO 9821683 to (“Touboul”).
<b>1014</b>	U.S. Patent No. 6,088,803 (“Tso”)
<b>1015</b>	U.S. Patent No. 5,889,943 (“Ji”)
<b>1016</b>	Li Gong <i>et al.</i> “Going Beyond the Sandbox: An Overview of the New Security Architecture in the Java Development Kit 1.2” (1997)
<b>1017</b>	Webpage: Oracle - Java Security Architect
<b>1018</b>	Paul Sabanal, Mark Yason, and Mark Vincent “Digging Deep Into the Flash Sandboxes” (2012)
<b>1019</b>	Webpage: Oracle - Deploying With the Applet Tag
<b>1020</b>	Youngang Song <i>et al.</i> “BRSS: A Binary Rewriting Security System for Mobile Code”
<b>1021</b>	Youngang Song and Brett D. Fleisch “Utilizing Binary Rewriting for Improving End-host Security” IEEE Vol. 18, No. 12 (Dec. 2007)
<b>1022</b>	Stephen McCamant and Greg Morrisett “Efficient, Verifiable Binary Sandboxing for CISC Architecture”

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

<b>Exhibit No.</b>	<b>Description of Document</b>
<b>1023</b>	Virus Bulletin (March 1991)
<b>1024</b>	Patent Application 11/159,455 Office Action – Non-Final Rejection
<b>1025</b>	Patent Application 11/159,455 – Patent Owner Amendment and Response to Office Action Under 37.C.F.R. §1.111
<b>1026</b>	Patent Application 11/159,455 - Notice of Allowance and Fee(s) due (May 26, 2009)
<b>1027</b>	90/013,016 Reexam Non-Final Office Action (November 19, 2013)
<b>1028</b>	90/013,016 Reexam Supplemental Amendment to Correct Priority Paragraphs Required by 37 CFR §§ 1.78 (August 25, 2014)
<b>1029</b>	90/013,016 Reexam Notice of Appeal (June 22, 2015)
<b>1030</b>	Patent Application 11/159,455 Data Sheet
<b>1031</b>	U.S. Pat. No. 6,804,780 to Touboul
<b>1032</b>	U.S. Pat. No. 6,480,962 to Touboul
<b>1033</b>	Plaintiff Finjan, Inc.'s Reply Claim Construction Brief, <i>Finjan, Inc. v. Blue Coat Systems, Inc.</i> , 13-cv-3999-BLF (July 7, 2014)
<b>1034</b>	Joint Post-Hearing Claim Construction Chart, Ex. A, <i>Finjan Software, Ltd. v. Secure Computing Corporation et al.</i> 06-cv-369-GMS (October 30, 2007)
<b>1035</b>	Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, <i>Finjan, Inc. v. Websense, Inc.</i> , 13-cv-4398-BLF (September 23, 2014.)
<b>1036</b>	Order Construing Claims, <i>Finjan, Inc. v. Blue Coat Systems, Inc.</i> , 13-cv-3999-BLF (October 20, 2014)
<b>1037</b>	Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, <i>Finjan, Inc. v. Proofpoint, Inc. and Armorize Technologies, Inc.</i> , 5:13-cv-5808-HSG (May 1, 2015)
<b>1038</b>	Claim Construction Order, <i>Finjan Software, Ltd. v. Secure Computing et al.</i> 06-cv-369-GMS (December 11, 2007)
<b>1039</b>	Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, <i>Finjan, Inc. v. Blue Coat Systems, Inc.</i> , 13-cv-3999-BLF (June 16, 2014)
<b>1040</b>	Provisional Application No. 60/205,591

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

<b>Exhibit No.</b>	<b>Description of Document</b>
<b>1041</b>	Mark Brown “ <i>Using Netscape 3,</i> ” (1996)
<b>1042</b>	90/013,016 Reexam Response to Non-Final Office Action (February 19, 2014)
<b>1043</b>	Finjan Investor Presentation, Q1 (2013)
<b>1044</b>	Dr. Frederick Cohen “Computer Viruses: Theory and Experiments” (1987)
<b>1045</b>	Thomas M. Chen and Jean-Marc Robert “The Evolution of Viruses and Worms”
<b>1046</b>	Virus Bulletin Issue Archive
<b>1047</b>	Sandeep Kumar and Eugene H. Spafford “A Generic Virus Scanner in C++,” (September 17, 1992)
<b>1048</b>	Morgan B. Adair “Detecting Viruses in the NetWare Environment”
<b>1049</b>	Virus Bulletin (November 1991)
<b>1050</b>	Virus Bulletin, (December 1991)
<b>1051</b>	Webpage: McAfee Antivirus product page
<b>1052</b>	Webpage: Norton Antivirus product page
<b>1053</b>	Webpage: Information Security StackExchange
<b>1054</b>	Webpage: W3Schools, JavaScript Tutorial page
<b>1055</b>	Sarah Gordon and David Chess “Attitude Adjustment: Trojans and Malware on the Internet: An Update”
<b>1056</b>	Andreas Moser <i>et al.</i> “Limits of Static Analysis for Malware Detection”
<b>1057</b>	Ian Goldberg “A Secure Environment for Untrusted Helper Applications (Confining the Wily Hacker)” (July 1996)
<b>1058</b>	Wayne A. Jansen “Countermeasures for Mobile Agent Security”
<b>1059</b>	Byron Cook <i>et al.</i> “Proving Program Termination,” Communications of the ACM, Vol. 54, No. 5 (May 2011)
<b>1060</b>	Webpage: Schneier on Security
<b>1061</b>	Javier Esparza “Decidability of Model Checking for Infinite-State Concurrent Systems”
<b>1062</b>	Edmund Clarke <i>et al.</i> “Model Checking and State Explosion Problem”

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

<b>Exhibit No.</b>	<b>Description of Document</b>
<b>1063</b>	Drew Dean <i>et al.</i> “Java Security: From HotJava to Netscape and Beyond”
<b>1064</b>	NSA Defense in Depth
<b>1065</b>	Dr. Thomas Porter “The Perils of Deep Packed Inspection”
<b>1066</b>	Mark J. Smith <i>et al.</i> “Protecting a Private Network: The AltaVista Firewall”
<b>1067</b>	Check Point Firewall-I™ White Paper, Version 3.0 (June 1997)
<b>1068</b>	Emin Gün Sirer <i>et al.</i> “Design and Implementation of a Distributed Virtual Machine for Networked Computers”
<b>1069</b>	Intrusion Detection Systems Group Test (Edition 2) – An NSS Group Report
<b>1070</b>	Dries Vanoverberghe and Frank Piessens “A Caller-Side Inline Reference Monitor for an Object-Oriented Intermediate Language”
<b>1071</b>	Ulfar Erlingsson “The Inlined Reference Monitor Approach to Security Policy Enforcement” (2004)
<b>1072</b>	Ari Luotonen and Kevin Altis “World-Wide Web Proxies” (April 1994)
<b>1073</b>	James Gosling and Henry McGilton “The Java™ Language Environment: A White Paper” (May 1996)
<b>1074</b>	Webpage: “A Simple Guide to HTML”
<b>1075</b>	David M. Martin Jr. <i>et al.</i> “Blocking Java Applets at the Firewall” (1997)
<b>1076</b>	Eric Perlman and Ian Kallen “Common Internet File Formats”
<b>1077</b>	“Developing Stored Procedures in Java: An Oracle Technical White Paper” (April 1999)
<b>1078</b>	Larry L. Peterson <i>et al.</i> “OS Support for General-Purpose Routers”
<b>1079</b>	Roel Wieringa “Traceability and Modularity in Software Design”
<b>1080</b>	U.S. Patent No. 6,434,499 (“Ulrich”)
<b>1081</b>	90/013,016 Reexam Renewed Petition to Accept Unintentionally Delayed Priority Claim Under 37 C.F.R. § 1.78
<b>1082</b>	2015-09-13 Declaration of Peter Kent in support of the “Brown” reference
<b>1083</b>	U.S. Patent No. 7,058,822 (“Edery”)

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

<b>Exhibit No.</b>	<b>Description of Document</b>
<b>1085</b>	Provisional Application No. 60/030,639
<b>1086</b>	U.S. Patent No. 6,092,194 (“Touboul”)
<b>1087</b>	U.S. Patent No. 6,167,520 (“Touboul”)
<b>1088</b>	2014-02-18 Phil Hartstein declaration in 90/013,016 Reexam
<b>1089</b>	90/013,016 Reexam Final Rejection (September 8, 2014)
<b>1090</b>	Webpage: Finjan Software Company Overview
<b>1091</b>	Excerpted Markman Hearing Transcript, Finjan, Inc. v. Blue Coat Systems, Inc., 13-cv-3999-BLF (August 22, 2014)
<b>1092</b>	Affidavit of Christopher Butler of the Internet Archive (“Brown”)
<b>1093</b>	Affidavit of David Sherfese of Alexa Internet
<b>1094</b>	U.S. Application No. 09/861,229
<b>1095</b>	Affidavit of Christopher Butler of the Internet Archive (“Shin”)

## **II. LEGAL PRINCIPLES USED IN THE ANALYSIS**

26. I am not a patent attorney, nor have I independently researched the law on patent validity. Attorneys for the Petitioner have explained certain legal principles to me that I have relied upon in forming my opinions set forth in this report.

### **A. Person Having Ordinary Skill in the Art (“POSA”)**

27. I understand that I must undertake my assessment of the claims of the ’633 patent from the perspective of what would have been known or understood by a POSA as of the earliest claimed priority date of the patent claim.

28. Counsel has advised me that to determine the appropriate level of one

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

of ordinary skill in the art, I may consider the following factors: (a) the types of problems encountered by those working in the field and prior art solutions thereto; (b) the sophistication of the technology in question, and the rapidity with which innovations occur in the field; (c) the educational level of active workers in the field; and (d) the educational level of the inventor.

29. The relevant technology field for the '633 patent is security programs, including content scanners for program code. Based on this, and the four factors above, it is my opinion that POSA would hold a bachelor's degree or the equivalent in computer science (or related academic fields) and three to four years of additional experience in the field of computer security, or equivalent work experience. This definition of the POSA would not change whether the time of the alleged invention is deemed to be 1997 or 2001.

30. Unless otherwise specified, when I mention a POSA or someone of ordinary skill, I am referring to someone with the above level of knowledge and understanding.

31. Based on my experiences, I have a good understanding of the capabilities of a person of ordinary skill in the relevant field. Indeed, in addition to being a person of at least ordinary skill in the art, I have worked closely with many such persons over the course of my career, and I have regularly taught material fundamental to the art in my role as professor and researcher over the past 22



Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

years.

**B. Prior Art**

32. I understand that the law provides categories of information that constitute prior art that may be used to anticipate or render obvious patent claims. To be prior art to a particular patent under the relevant law, a reference must have been made, known used, published, or patented, or be the subject of a patent application by another, before the priority date of the patent. I also understand that the POSA is presumed to have knowledge of the relevant prior art.

33. As discussed below, I understand that the Petitioner has determined that various claims of the '633 patent are entitled to different priority dates. These dates range from 2000 to 2001. However, other than the differences in which art is considered prior art, my conclusions and discussion in this declaration would not be substantively different regardless of which date in that range is ascribed to the POSA.

**C. Broadest Reasonable Interpretations**

34. I understand that, in *Inter Partes* Review, the claim terms are to be given their broadest reasonable interpretation (BRI) in light of the specification. See 37 C.F.R. § 42.100(b). In performing my analysis and rendering my opinions, I have interpreted claim terms for which the Petitioner has not proposed a BRI construction by giving them the ordinary meaning they would have to a POSA,

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

reading the '633 patent with its effective filing date in mind (May 17, 2000, or May 17, 2001, depending on the particular petitioned claim at issue), and in light of its specification and file history.

**D. Standards for Anticipation and Obviousness**

35. I understand that 35 U.S.C. §§ 102 and 103 contain a variety of requirements for obtaining a patent. I understand that pursuant to those sections, a patent is invalid if it was anticipated or rendered obvious by the prior art.

36. I understand that the Model Patent Jury Instructions for the Northern District of California (June 17, 2014) provide the following instructions for anticipation and obviousness:

4.3a1 ANTICIPATION

A patent claim is invalid if the claimed invention is not new. For the claim to be invalid because it is not new, all of its requirements must have existed in a single device or method that predates the claimed invention, or must have been described in a single previous publication or patent that predates the claimed invention. In patent law, these previous devices, methods, publications or patents are called “prior art references.” If a patent claim is not new we say it is “anticipated” by a prior art reference. The description in the written reference does not have to be in the same words as the claim, but all of the requirements of the claim must be there, either stated or

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

necessarily implied, so that someone of ordinary skill in the field of [identify field] looking at that one reference would be able to make and use the claimed invention.

Here is a list of the ways that [alleged infringer] can show that a patent claim was not new [use those that apply to this case]:

[– if the claimed invention was already publicly known or publicly used by others in the United States before [insert date of conception unless at issue];]

[– if the claimed invention was already patented or described in a printed publication anywhere in the world before [insert date of conception unless at issue]. [A reference is a “printed publication” if it is accessible to those interested in the field, even if it is difficult to find.];]

[– if the claimed invention was already made by someone else in the United States before [insert date of conception unless in issue], if that other person had not abandoned the invention or kept it secret;]

[– if the claimed invention was already described in another issued U.S. patent or published U.S. patent application that was based on a patent application filed before [insert date of the patent holder’s application filing date] [or] [insert date of conception unless at issue];]

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

[– if [named inventor] did not invent the claimed invention but instead learned of the claimed invention from someone else;]

[– if the [patent holder] and [alleged infringer] dispute who is a first inventor, the person who first conceived of the claimed invention and first reduced it to practice is the first inventor. If one person conceived of the claimed invention first, but reduced to practice second, that person is the first inventor only if that person (a) began to reduce the claimed invention to practice before the other party conceived of it and (b) continued to work diligently to reduce it to practice. [A claimed invention is “reduced to practice” when it has been tested sufficiently to show that it will work for its intended purpose or when it is fully described in a patent application filed with the PTO].]

[Since it is in dispute, you must determine a date of conception for the [claimed invention] [and/or] [prior invention]. Conception is the mental part of an inventive act and is proven when the invention is shown in its complete form by drawings, disclosure to another or other forms of evidence presented at trial.]

(Model Patent Jury Instructions for the Northern District of California at 30-31, § 4.3a1 (June 17, 2014).)

4.3a2 STATUTORY BARS

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

A patent claim is invalid if the patent application was not filed within the time required by law. This is called a “statutory bar.” For a patent claim to be invalid by a statutory bar, all of its requirements must have been present in one prior art reference dated more than one year before the patent application was filed. Here is a list of ways [alleged infringer] can show that the patent application was not timely filed: [choose those that apply]

[– if the claimed invention was already patented or described in a printed publication anywhere in the world before [insert date that is one year before effective filing date of patent application]. [A reference is a “printed publication” if it is accessible to those interested in the field, even if it is difficult to find.];]

[– if the claimed invention was already being openly used in the United States before [insert date that is one year before application filing date] and that use was not primarily an experimental use (a) controlled by the inventor, and (b) to test whether the invention worked for its intended purpose;]

[– if a device or method using the claimed invention was sold or offered for sale in the United States, and that claimed invention was ready for patenting, before [insert date that is one year before application filing date]. [The claimed invention is not being [sold] [or] [offered for

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

sale] if the [patent holder] shows that the [sale] [or] [offer for sale] was primarily experimental.] [The claimed invention is ready for patenting if it was actually built, or if the inventor had prepared drawings or other descriptions of the claimed invention that were sufficiently detailed to enable a person of ordinary skill in the field to make and use the invention based on them.];]

[– if the [patent holder] had already obtained a patent on the claimed invention in a foreign country before filing the original U.S. application, and the foreign application was filed at least one year before the U.S. application.]

For a claim to be invalid because of a statutory bar, all of the claimed requirements must have been either (1) disclosed in a single prior art reference, (2) implicitly disclosed in a reference to one skilled in the field, or (3) must have been present in the reference, whether or not that was understood at the time. The disclosure in a reference does not have to be in the same words as the claim, but all the requirements must be there, either described in enough detail or necessarily implied, to enable someone of ordinary skill in the field of [identify field] looking at the reference to make and use the claimed invention.

(Model Patent Jury Instructions for the Northern District of California at 32, § 4.3a2 (June 17, 2014).)

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

#### 4.3b OBVIOUSNESS – (Alternative 1)

Not all innovations are patentable. A patent claim is invalid if the claimed invention would have been obvious to a person of ordinary skill in the field [at the time the application was filed][as of [insert date]]. The court, however, is charged with the responsibility of making the determination as to whether a patent claim was obvious based upon your determination of several factual questions.

First, you must decide the level of ordinary skill in the field that someone would have had [at the time the claimed invention was made] [as of the effective filing date of the claimed invention]. In deciding the level of ordinary skill, you should consider all the evidence introduced at trial, including:

- (1) the levels of education and experience of persons working in the field;
- (2) the types of problems encountered in the field; and
- (3) the sophistication of the technology.

[Patent holder] contends that the level of ordinary skill in the field was [ ]. [Alleged infringer] contends that the level of ordinary skill in the field was [ ].

Second, you must decide the scope and content of the prior art. [Patent holder] and [alleged infringer] disagree as to whether [identify prior art reference(s)] should be

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

included in the prior art you use to decide the validity of claims [ ] of the [ ] patent. In order to be considered as prior art to the [ ] patent, these references must be reasonably related to the claimed invention of that patent. A reference is reasonably related if it is in the same field as the claimed invention or is from another field to which a person of ordinary skill in the field would look to solve a known problem.

Third, you must decide what difference, if any, existed between the claimed invention and the prior art.

Finally, you must determine which, if any, of the following factors have been established by the evidence:

[(1) commercial success of a product due to the merits of the claimed invention];]

[(2) a long felt need for the solution provided by the claimed invention];]

[(3) unsuccessful attempts by others to find the solution provided by the claimed invention[;]

[(4) copying of the claimed invention by others];]

[(5) unexpected and superior results from the claimed invention]]

[(6) acceptance by others of the claimed invention as shown by praise from others in the field or from the licensing of the claimed invention];]

[(7) other evidence tending to show nonobviousness];]



Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

[(8) independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it]; and]

[(9) other evidence tending to show obviousness].]

(Model Patent Jury Instructions for the Northern District of California at 34-35, § 4.3b (June 17, 2014).)

#### 4.3b OBVIOUSNESS – (Alternative 2)

Not all innovations are patentable. A patent claim is invalid if the claimed invention would have been obvious to a person of ordinary skill in the field [at the time the claimed invention was made] [as of the effective filing date of the claimed invention]18 [as of [insert date]].

This means that even if all of the requirements of the claim cannot be found in a single prior art reference that would anticipate the claim or constitute a statutory bar to that claim, a person of ordinary skill in the field of [identify field] who knew about all this prior art would have come up with the claimed invention.

The ultimate conclusion of whether a claim is obvious should be based upon your determination of several factual decisions.

First, you must decide the level of ordinary skill in the field that someone would have had [at the time the claimed invention was made] [as of the effective filing date of the claimed invention]. In deciding the level of

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

ordinary skill, you should consider all the evidence introduced at trial, including:

- (1) the levels of education and experience of persons working in the field;
- (2) the types of problems encountered in the field; and
- (3) the sophistication of the technology.

[Patent holder] contends that the level of ordinary skill in the field was [ ]. [Alleged infringer] contends that the level of ordinary skill in the field was [ ].

Second, you must decide the scope and content of the prior art. [Patent holder] and [alleged infringer] disagree as to whether [identify prior art reference(s)] should be included in the prior art you use to decide the validity of claims [ ] of the [ ] patent. In order to be considered as prior art to the [ ] patent, these references must be reasonably related to the claimed invention of that patent. A reference is reasonably related if it is in the same field as the claimed invention or is from another field to which a person of ordinary skill in the field would look to solve a known problem.

Third, you must decide what difference, if any, existed between the claimed invention and the prior art.

Finally, you should consider any of the following factors that you find have been shown by the evidence:

[(1) commercial success of a product due to the merits of the claimed invention];]

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

[(2) a long felt need for the solution provided by the claimed invention];]

[(3) unsuccessful attempts by others to find the solution provided by the claimed invention];]

[(4) copying of the claimed invention by others];]

[(5) unexpected and superior results from the claimed invention];]

[(6) acceptance by others of the claimed invention as shown by praise from others in the field or from the licensing of the claimed invention];]

[(7) other evidence tending to show nonobviousness];]

[(8) independent invention of the claimed invention by others before or at about the same time as the named inventor thought of it] [; and]

[(9) other evidence tending to show obviousness].]

[The presence of any of the [list factors 1-7 as appropriate] may be considered by you as an indication that the claimed invention would not have been obvious [at the time the claimed invention was made] [as of the effective filing date of the claimed invention], and the presence of the [list factors 8-9 as appropriate] may be considered by you as an indication that the claimed invention would have been obvious at such time.

Although you should consider any evidence of these factors, the relevance and importance of any of them to

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

your decision on whether the claimed invention would have been obvious is up to you.]

A patent claim composed of several elements is not proved obvious merely by demonstrating that each of its elements was independently known in the prior art. In evaluating whether such a claim would have been obvious, you may consider whether [the alleged infringer] has identified a reason that would have prompted a person of ordinary skill in the field to combine the elements or concepts from the prior art in the same way as in the claimed invention. There is no single way to define the line between true inventiveness on the one hand (which is patentable) and the application of common sense and ordinary skill to solve a problem on the other hand (which is not patentable). For example, market forces or other design incentives may be what produced a change, rather than true inventiveness. You may consider whether the change was merely the predictable result of using prior art elements according to their known functions, or whether it was the result of true inventiveness. You may also consider whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent. Also, you may consider whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way. You may

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

also consider whether the claimed invention would have been obvious to try, meaning that the claimed innovation was one of a relatively small number of possible approaches to the problem with a reasonable expectation of success by those skilled in the art. However, you must be careful not to determine obviousness using the benefit of hindsight; many true inventions might seem obvious after the fact. You should put yourself in the position of a person of ordinary skill in the field [at the time the claimed invention was made] [as of the effective filing date of the claimed invention] and you should not consider what is known today or what is learned from the teaching of the patent.

(Model Patent Jury Instructions for the Northern District of California at 36-38, § 4.3b (June 17, 2014).)

37. I am also informed that the United States Patent Office supplies its examining corps with a *Manual of Patent Examining Procedure* that provides exemplary rationales that may support a conclusion of obviousness, including:

- (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 known technique to improve similar devices (methods, or products) in the same way;

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

(D) Applying a known technique to a known device (method, or product) ready for improvement to yield predictable results;

(E) “Obvious to try” – choosing from a finite number of identified, predictable solutions, with a reasonable expectation of success;

(F) Known work in one field of endeavor may prompt variations of it for use in either the same field or a different one based on design incentives or other market forces if the variations are predictable to one of ordinary skill in the art;

(G) Some teaching, suggestion, or motivation in the prior art that would have led one of ordinary skill to modify the prior art reference or to combine prior art reference teachings to arrive at the claimed invention.

(MPEP § 2143(I).)

### **III. TECHNOLOGY TUTORIAL**

38. A computer program is a sequence of instructions that tell a computer processor what to do. Computer processors will not question the instructions they are given no matter how harmful these instructions may be. And processors never tire—most can process millions or billions of instructions per second and can continue processing indefinitely, whether or not a human is observing the processing.

39. Although processors can be monitored, the sequence of computer instructions in a software program is generally too large, too complicated, and too cryptic for a human to easily review. Even if these instructions could be manually

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

inspected, it would be very difficult to ensure that they had not been modified by another program at some point in time before they reached the processor.

40. Generally speaking, security professionals refer to any kind of harmful program as “malware.” Viruses, a subtype of malware, work by inserting harmful instructions that hijack the execution flow of a normal program. At some point while the processor is executing the infected program, it reaches the inserted instructions and the viral code takes over. The virus generally performs some kind of self-replication by, inserting the viral code into other programs, along with other functions. (Ex. 1044 at 2, col. 2, to 3, col. 2; Ex. 1045 at 1-2.)

41. In addition to viruses, other types of malware include Trojan horse, worm, and so forth. These distinctions are generally related to how the malware “broke in” or how it spread. These kinds of issues are the purview of the security practitioner. The consumer, for the most part, does not care how the bad guy deleted their photos of their kids or how it got to their computer. For these reasons, the most common and most recognized type of malware, a virus, is often used as a placeholder for malware of any kind.

42. It is important to understand that, unfortunately, there will never be an end to malware. Computer scientists have proven that it is impossible to find a general solution to this problem. This means that there will never be a perfect method for detecting any and all harmful instructions. (Ex. 1044 at 7, col. 2; Ex.

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

1045 at 13-14.) Or, said another way, it is impossible to write a program that can recognize all possible harmful instructions. As soon as one security measure is implemented, someone else will find a way around it.

43. Accordingly, those individuals and organizations that produce harmful computer instructions and those that stop them have been and will always be in an arms race. Each year, the “bad guys” will find new ways to get processors to do things they should not and each year the “good guys” will have to find new ways to defeat them. This ongoing struggle was already in full swing by the beginning of the 1990’s as the anti-virus community matured both in academic research and commercial products. (*See, e.g.*, Ex. 1046 at 1; Ex. 1047 at 4-5, 12; Ex. 1048 at 1-2, 7.)

44. However, the “arms race” does not mean that every update is revolutionary. Some concepts have enjoyed significant longevity and are simply updated to remain current. For example, one of the earliest approaches to defeating viruses and other kinds of malware is still used today: scanners.

45. The basic concept behind a scanner is to search for known patterns and/or sequences of malware in computer resources. (Ex. 1023 at 7-9.) Early scanners emerged in 1988 and as late as 1991 most viruses could be detected by searching for an exact sequence of bytes. Each of these kinds of virus has a specific sequence of code that could, with relative ease, be recognized. (Ex. 1023



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

at 7-9; Ex. 1049 at 5-7.) Nevertheless, antivirus researchers already could see the next generation of viruses on the horizon that would be harder to detect, either because the virus would be obscured or encrypted, or because the virus would have multiple forms. (Ex. 1049 at 19; Ex. 1050 at 15-16.)

46. In the later 1990's, stopping malware became much more difficult because of the explosive growth of the Internet. Prior to extensive use of email and web-browsing, the most common way to get a virus was through software on a shared diskette. For this reason, (wise) users were very careful about which friends they accepted disks from and scanned even these disks before accepting new content onto their system.

47. But with the advent of email and browsing, users suddenly had their computers connected to dozens (or even hundreds) of sources on a daily basis (every email from a unique sender constitutes a different source). (Ex. 1010 at 5-6.)

48. Worse, perhaps, was the introduction of “mobile code”—code transmitted to and executed by a client without explicit user intervention. When copying a file from a diskette, a user was at very low risk of infection unless they *executed* the file. But web browsers (or associated software) began providing ways for the content they received to execute instructions on the local computer without any explicit user-directed execution step. In other words, the user's

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

computer can execute instructions from untrusted remote locations *automatically*.

(Ex. 1011 at 8.)

49. With this new context came new approaches to security. Although the scanners from the 1990's continue today (although considerably more updated and powerful and with a variety of new technologies for evaluating the trustworthiness of mobile code) (Ex. 1051, Ex. 1052), the original technologies upon which these products were founded would be ineffective in protecting users against malicious mobile code. Mobile code is simply too varied, too extensive, and too dynamic for the same kind of analysis that is useful for protecting users against traditional viruses and malware. (Ex. 1011 at 6-8; Ex. 1053 at 2.)

50. As a way of illustrating the problem, many websites use code written in languages like JavaScript to make their pages dynamic. JavaScript is known as a language that web developers "must" learn due to its prevalence. (Ex. 1054 at 5.) JavaScript is a programming language that most browsers are designed to support. That means that when web browsers download a webpage with JavaScript, the browser executes the JavaScript instructions. JavaScript allows pages to be interactive with the user; otherwise, the content of the page could not change without reloading.

51. Security concepts from the early 90's are not a good match for languages like JavaScript. Each site's JavaScript code is generally customized

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

and may experience regular or updates in order to provide new functionality or a better user experience. Considering the length of time necessary to do a traditional signature-based anti-virus scan, there is no way to scan each page as it is received by the browser without significantly and drastically reducing performance. Moreover, given how extensively users browse the web, new malware in webpages would do catastrophic amounts of damage before updated signatures, or malicious code fingerprints, could be released to prevent further destruction.

52. As mentioned above, there is a fundamental theoretical problem: no computer program can accurately evaluate all other computer programs for malware. That is why scanners must rely on signatures for so much of their anti-malware strategy. Unfortunately, signatures are just too inflexible for dealing with a sea of dynamic mobile code. (Ex. 1055 at 13-14; Ex. 1056 at Abstract, 1-2.) Because users had little hope of knowing which mobile code to trust, the solution was to make mobile code harmless. Absolutely fundamental, therefore, to the security realities of the Internet in the mid-to-late 1990s, was the sandbox.

53. Basically, a sandbox, as used in the general computer industry, is any mechanism that prevents the executing code from accessing anything that does damage. (Ex. 1057 at 4, col. 1.) Most code is really not capable of doing much harm to the underlying host system. Much of the time it is the “system

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

calls” that some code contains which are dangerous (calls that ask the operating system to do things like write to a file, read from memory, or execute a program). By denying mobile code access to dangerous system calls, it (theoretically) doesn’t matter what other operations the code performs. The restricted environment in such schemes is termed “a sandbox.” There are a lot of different methods and mechanisms for sandboxing code. (*See* Ex. 1018 at 1, 6-50.)

54. The Java programming language was designed for sandbox operations from the very beginning. (Ex. 1016 at 2-3; Ex. 1017 at 1-2.) Java programs do not get executed by the processor directly. (Ex. 1016 at 2-3; Ex. 1017 at 1.) Instead, every computer that needs to run Java must have a “Java Virtual Machine” (JVM). (Ex. 1017 at 1.) A JVM is like a processor simulated in software. The Java program is executed by the JVM instead of the processor. (Ex. 1016 at 2-3; Ex. 1017 at 1.)

55. Java programs do not invoke system calls directly. All system calls are processed by the JVM first. (Ex. 1016 at 2-3; Ex. 1017 at 1.) From the beginning, Java included a “Security Manager” that determines whether a Java system call is authorized. (Ex. 1017 at 1-2 (“[A]ccess to crucial system resources is mediated by the Java Virtual Machine and is checked in advance by a Security Manager class that restricts the actions of a piece of untrusted code to the bare minimum.”); Ex. 1016 at 2-3.) From this perspective, all Java programs are run in

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

a “sandboxed” or restricted environment. (Ex. 1017 at 1-2; Ex. 1016 at 2-3.)

56. However, Java distinguishes between two types of Java programs. (Ex. 1016 at Fig. 1.) A Java program can be run directly on the computer by a user. These programs (generally called “Java applications”) are, unless otherwise configured, executed with full permissions. (*See id.* at 5, Fig. 1.) So, even though they are running in the JVM, most technical users would not consider them sandboxed because they are allowed to do pretty much anything they want. (*Id.*)

57. On the other hand, Java provides a distinct class of Java programs explicitly designed for use over the Internet (mobile code). These programs, called “Applets,” are obtained from the open network and can be run directly within a browser, or as a separate window launched from a browser. (*Id.* at 1-3.)

58. When the JVM executes an Applet, it enforces a large number of rules and restrictions to make sure that the mobile code cannot do bad things. These rules and restrictions are typically what are thought of when one speaks of the “Java Sandbox.” (Ex. 1016 at 2-3; Ex. 1017 at 1.) By way of example, Applets are prohibited from accessing the file system and have strict limits on network accessibility.

59. Sandboxing is also used by other Mobile Code systems (e.g., the “padded cell” approach to Safe-TCL and Mozilla’s implementation of JavaScript). (Ex. 1058 at 4-5.)

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

60. While sandboxes are important, even critically essential, to the safe operation of mobile code, they are not foolproof. Computer scientists have demonstrated that it is impossible to determine if any given computer program is “correct” in the general case. In recent years, our understanding of mathematics and our computational tools have improved to where many practical programs can be proved “correct,” but most programs are still too complex, and even the definitions of “correctness” too difficult to specify, to *prove* that they will work as intended and without bugs. (Ex. 1059 at 1; Ex. 1060 at 1; Ex. 1061 at 2, 21-22; Ex. 1062 at 8-11.) If there are defects in the sandbox, malicious entities can often find a way to exploit those defects to do damage to the user’s system. Sometimes, there is simply a flaw in design and permitted operations that were thought to be harmless are not. This could be due to a combination of operations, but may also be simply because the operation is used in a way that was unintended. Other times, the flaw is actually in the implementation of the sandbox in the form of a bug or other error. When malware can exploit some bug or hole to gain privileges normally denied by the sandbox it is often called “breaking out of the sandbox.” (Ex. 1018 at 1, 59.)

61. As a side note, in my instruction at Johns Hopkins, I have my students both create their own sandboxes for mobile code and look for exploits in their classmates’ sandboxes. It is an amazingly difficult process to create a

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

sandbox with sufficient power to get necessary work done while also blocking harmful operations. Moreover, it is very difficult to create the sandbox without exploitable bugs.

62. Almost immediately after Java was introduced, security researchers found significant vulnerabilities in the Java sandbox for Applets. (Ex. 1063 at 1, 5-10.) These flaws are design-level flaws (there are multiple implementations of the JVM) that permitted Applets to either escape the sandbox or make use of permitted operations in harmful ways. (*Id.*) The design of the JVM and the sandbox has required a number of updates since its introduction in order to plug the various holes that are uncovered. (*Id.*)

63. Because of the potential failures in sandboxes, entities concerned about network security often employ multiple layers of security in their systems. Sometimes called “Defense in Depth” the idea is that by layering security at multiple levels, damage can be mitigated or prevented completely when security defenses fail. (Ex. 1064 at 1-2.)

64. Consider, for example, that virus definitions on an individual computer may be out of date; perhaps the user is not keeping his definitions current. Or, worse, perhaps a corporation has security policies that should be in place on every computer but, for whatever reason, are not enforced in one or more nodes. One potential solution is to place an additional layer of security at

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

the firewall or gateway.

65. Gateway computers are as old as the Internet Protocol itself. (Ex. 1012 at 1-12.) Because the Internet Protocol is designed to interconnect networks, the node that provides this inter-connectivity is called the “gateway.” (*Id.* at 1-12.) All the other computers on the smaller network (usually a Local Area Network (LAN)), funnel traffic meant for other networks through the gateway computer.

66. During the 1990’s, security professionals and researchers realized that gateway computers made excellent systems for enforcing security because most organizations running a computer network place higher trust in the local computers than in the external computers. As a result of the difference in security profiles for intranet versus extranet computers, the gateway separating the two is a natural location to enforce policies. Specifically, network activity that may be trusted amongst the local computers can be blocked and prevented from traffic entering the local network from beyond the gateway.

67. Early firewalls simply blocked specific IP addresses; later models could discriminate on ports. Certain logical ports are associated with different traffic: port 80 for web traffic, port 25 for email, etc. Firewalls could block all traffic on ports not explicitly allowed to provide service. (Ex. 1065 at 1-2.)

68. As firewalls became more complicated, they could even start to



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

monitor the traffic itself. For example, as early as 1997, the Alta Vista firewall could monitor FTP traffic with such granularity as to allow “GET” requests but not “PUT” requests (i.e., it could enforce allowing downloads over FTP but block uploads). (Ex. 1066 at 8.) It did this by actually reading the entire traffic stream, decoding it, and monitoring the specific application layer protocol operations. (*Id.*)

69. The Checkpoint FireWall 1, also in 1997, provided anti-virus scanning at the firewall/gateway itself. (Ex. 1067 at 4, 19, 21.) By putting anti-virus scanning at the gateway, an enterprise could ensure up-to-date checking of all content as it entered the local network. (*Id.*) Such scanning could not replace anti-virus scanning on the host computer; rather, it was meant to complement it by providing an additional layer of security. (*Id.*) IT professionals now realized that there was another reason for gateway security systems: centralization of policy and policy enforcement. (Ex. 1068 at 4, col. 1.)

70. Additional mechanisms for network defense were also added in the late 1990’s. In particular, Intrusion Detection Systems (IDS) were widely researched and multiple commercial products were developed. (Ex. 1069 at 11-19.) As an additional layer of security, these systems were rarely designed to actively stop attacks. Instead, they monitored network traffic for anomalies in order to detect new attacks for which no signature yet existed. (*Id.*) Some IDS

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

systems ran on host computers, others on the network, and some at the Gateway.

(*Id.*)

71. Yet another layer of defense specifically for mobile code was to perform “code rewriting” or “code instrumentation” in order to insert additional protection code. (Ex. 1020 at 1.) When the code is downloaded from the Internet, it can be re-written at certain well-defined locations (such as function definitions and function call cites) with additional code to protect the system. One might, for example, insert a call at the beginning of each function to determine if processor usage is within a threshold. If not, the function could be forced to terminate early. This kind of monitoring code is sometimes called an “inline reference monitor.” (Ex. 1070 at 1.)

72. By way of further explanation, a function is, in many ways, similar to a “mini program.” Programmers divide up the code into functions for better organization, code-reuse, and easier readability. Within the larger program, a function is a chunk of code that has a name used to execute it. One function typically calls (executes) a number of other functions. Each function is defined before it is used and the definition includes a name and the code that constitutes the body of the function.

73. It is relatively easy to add a layer of security to a function by inserting code right at the very beginning of the function body. Every time that

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

function is called the newly inserted code will be executed first.

74. Certain forms of code rewriting are another form of sandboxing because they prevent a function from being able to do harmful activities. (Ex. 1022 at 1-2.) One example might be rewriting the code so that a function that creates a graphical window cannot be called if a window is already open (preventing mobile code that spawns infinite windows on a user's desktop). The initial instructions in the function check if a window is already open. If so, it terminates the function with an error. (*See* Ex. 1071 at 19-21.)

#### **IV. THE '633 PATENT**

##### **A. Overview of the '633 Patent**

75. The '633 patent recites claims for receiving downloadable-information, determining whether the downloadable-information includes executable code, and if it does, transmitting a Downloadable, mobile protection code, and/or security policies to the destination computer. (Ex. 1001 at Abstract, 2:39-57.)

76. The '633 recites various methods for determining whether the downloadable-information includes executable code, including analysis based on executable code characteristics. (Ex. 1001 at 19:18-47.) For example, claims 4, 6 and 7 recite analysis methods that, respectively, focus on type indicators and information patterns characteristic of executable code.

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

77. The '633 patent also claims a system and method for creating a sandboxed package or protective environment, by packaging together a Downloadable, mobile protection code, and protection policies to be sent to and executed by the client. (Ex. 1001 at 3:5-21.) Execution of the mobile protection code and Downloadable ensure that potentially malicious actions of the Downloadable will be responded to in accordance with the attached protection policies. (Ex. 1001 at Abstract, 4:28-41.)

### **B. The Claims of the '633 Patent**

78. I will refer to the various elements of the claims of the '633 patent as identified below:

<b>Element</b>	<b>Claim Limitation</b>
1[a]	A computer processor-based method, comprising:
1[b]	receiving, by a computer, downloadable-information;
1[c]	determining, by the computer, whether the downloadable information includes executable code;
1[d]	based upon the determination, transmitting from the computer mobile protection code to at least one information destination of the downloadable-information, if the downloadable-information is determined to include executable code.
2	The method of claim 1, wherein the receiving includes monitoring received information of an information re-communicator.
3	The method of claim 2, wherein the information recommunicator is a network server.
4	The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included type indicator indicating an executable file type.
6	The method of claim 1, wherein the determining comprises analyzing the downloadable-information for an included file type indicator and an information pattern corresponding to one or more information patterns

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

	that tend to be included within executable code.
7	The method of claim 1, further comprising receiving, by the computer, one or more executable code characteristics of executable code that is capable of being executed by the information-destination, and wherein the determining is conducted in accordance with the executable code characteristics.
8[a]	A computer processor-based system for computer security, the system comprising
8[b]	an information monitor for receiving downloadable-information
8[c]	a content inspection engine communicatively coupled to the information monitor for determining, by the computer, whether the downloadable-information includes executable code; and
8[d]	a protection agent engine communicatively coupled to the content inspection engine for causing mobile protection code ("MPC") to be communicated by the computer to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code.
13[a]	A processor-based system for computer security, the system comprising:
13[b]	means for receiving downloadable-information;
13[c]	means for determining whether the downloadable-information includes executable code;
13[d]	means for causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code
14[a]	A computer program product, comprising a computer usable medium having a computer readable program code therein, the computer readable program code adapted to be executed for computer security, the method comprising:
14[b]	providing a system, wherein the system comprises distinct software modules, and
14[c]	wherein the distinct software modules comprise an information re-communicator and a mobile code executor;
14[d]	receiving, at the information re-communicator, downloadable-information including executable code;
14[e]	causing mobile protection code to be executed by the mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted,

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

	will be processed by the mobile protection code.
19	The method of claim 14, wherein the re-communicator is at least one of a firewall and a network server.
28[a]	A processor-based method, comprising:
28[b]	receiving a sandboxed package that includes mobile protection code ("MPC") and a Downloadable and one or more protection policies at a computer at a Downloadable-destination;
28[c]	causing, by the MPC on the computer, one or more operations attempted by the Downloadable to be received by the MPC;
28[d]	receiving, by the MPC on the computer, an attempted operation of the Downloadable; and
28[e]	initiating, by the MPC on the computer, a protection policy corresponding to the attempted operation.
34[a]	A processor-based system for computer security, the system comprising:
34[b]	a mobile code executor on a computer for initiating received mobile code;
34[c]	a sandboxed package capable of being received and initiated by the mobile code executor on the computer, the sandboxed package including a Downloadable and mobile protection code ("MPC") for causing one or more Downloadable operations to be intercepted by the computer and for processing the intercepted operations by the computer, if the Downloadable attempts to initiate the operations.

### C. Interpretation of Claim Limitations in the '633 Patent

#### 1. Non-Means-Plus-Function Limitations

79. I have identified broadest reasonable interpretations for certain claim terms in the following table:

Term	Broadest Reasonable Interpretation (BRI)
"mobile protection code ("MPC")"	"code that, at runtime, monitors or intercepts actually or potentially malicious code"
"information re-communicator"	"server"

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

80. My analysis in this declaration assumes that the terms in the above table are defined using the associated BRIs. From my reading of the '633 patent, I believe that these BRIs are consistent with how one of ordinary skill in the art at the time the '633 patent was filed (i.e., the effective filing date) would interpret the claim terms.

81. Regarding the term “mobile protection code,” the '633 patent describes mobile protection code as code that can detect, intercept, respond to, divert, filter, or log malicious actions. (Ex. 1001 at 10:2-4, 2:51-57, 3:7-10, 3:18-21.) For example, the '633 patent specification states that “[t]he sandboxed package includes mobile protection code (‘MPC’) for causing one or more predetermined malicious operations or operation combinations of a Downloadable to be monitored or otherwise intercepted.” (Ex. 1001 at 3:7-11.) The inclusion of “at runtime” in the BRI of “mobile protection code” is consistent with the specification, including the title, of the '633 patent (“Malicious Mobile Code Runtime Monitoring System and Methods”). (*See also* Ex. 1001 at 5:30-39.) Counsel has informed me that, in the Blue Coat litigation, Patent Owner argued that “mobile protection code” can include modifying a Downloadable’s executable code. (Ex. 1033 at 12:3-9; Ex. 1091 at 5:22- 8:6; Ex. 1039 at 17:13-19.)

82. Counsel has informed me that the Court in the Blue Coat litigation construed “mobile protection code” as “code that, at runtime, monitors or

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

intercepts actually or potentially malicious code operations.” (Ex. 1036 at 5-8.)

Counsel has also informed me that Patent Owner has indicated that it does not oppose the inclusion of “actually or potentially” in the construction of this term.

(Ex. 1091 at 3:6-21.)

83. The prior art references on which Petitioner relies disclose runtime monitoring and intervention. (*See, e.g.*, Ex. 1004 at 6, col. 1 (“*Once the applet is invoked*, AppletTrap’s monitoring code extracts information about the resources that will be used by the applet and ascertains the permissibility of this action by comparing it with the attached security policy”) (emphasis added); Ex. 1009 at 2 (“In this paper we propose a technique called bytecode modification, through which we put restrictions on applets by inserting additional bytecode instructions that will perform the necessary *run-time* tests”) (emphasis added).)

84. Regarding “information re-communicator,” the ’633 patent specification explains that this term encompasses various types of servers, including firewalls, resource servers, gateways, email relays, and other types of servers. (Ex. 1001 at 5:34–37 (“Embodiments provide, within one or more ‘servers’ (e.g. firewalls, resources, gateways, email relays or other information re-communicating devices), for receiving downloadable-information...”); *id.* at 2:58–62 (identifying “network connectable information re-communicating devices” as “‘servers’ or ‘re-communicators.’”); *see also id.* at 7:23–28



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

(identifying server 142b in FIG. 1c as a “re-communicator”); *Id.* at FIG. 9 (showing a “Monitor re-communicator (e.g. server)”.)

## 2. Means-Plus-Function Limitations

85. I understand that claim limitations presented in a means-plus-function format are construed in a two-step process, where the function associated with the limitation is first identified and then the structure corresponding to that function is identified.

86. My determinations regarding the functions and corresponding structures of the means-plus-function terms recited in the Petitioned Claims are reflected in the following table:

Term	Corresponding Structure & Function
“means for receiving downloadable-information”	Function: receiving downloadable-information Corresponding Structure: server, firewall, or information monitor
“means for determining whether the downloadable-information includes executable code”	Function: determining whether the downloadable-information includes executable code Corresponding Structure: code detection engine
“means for causing mobile protection code to be communicated to at least one information-destination of the downloadable-information”	Function: causing mobile protection code to be communicated to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code Corresponding Structure: packaging engine

87. From my reading of the ’633 patent, I believe that these

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

interpretations are consistent with how one of ordinary skill in the art at the time the '633 patent was filed (i.e., its effective filing date) would interpret the claim limitations.

88. Regarding the limitation “means for receiving downloadable-information,” the '633 patent specification identifies servers, firewalls, and “information monitors” as structures that receive downloadable-information. (Ex. 1001 at 9:10-13 (“[I]nformation received by server 301 (or firewall 302) can include non-executable information, executable information or a combination of non-executable and one more executable code portions”; *id.* at claim 8 (“[A]n information monitor for receiving downloadable information”).)

89. Regarding the limitation “means for determining . . . executable code,” the '633 patent identifies a code detection engine within a protection engine as a structure that determines whether downloadable-information includes executable code: “The protection engine includes . . . a code detection engine for determining whether the received information includes executable code.” (Ex. 1001 at 2:62–66; *see also id.* at FIG. 4 (showing a detection engine 402 including a code detector 421).) The protection engine and its components, including the code detection engine, are hosted by a server or a firewall, distributed among several different servers or firewalls, or on a standalone device:

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

[A] server 141b or firewall 143 can operate as a suitable protection engine host. A protection engine can also be implemented in a more distributed manner among two or more protection engine host systems or host system elements, such as both of server 141b and firewall 143, or in a more integrated-manner, for example, as a standalone device.

(*Id.* at 7:26–33.)

90. The protection engine and its components, including the code detection engine, can be implemented in hardware, software, or a combination of hardware and software. (Ex. 1001 at 7:66–8:2 (“[S]ystem 100 elements (FIGS. 1a-c) [including protection engines 142a–142c] are implemented in hardware, software or some combination by one or more processing systems consistent therewith . . . .”))

91. Regarding the “means for causing . . . .,” the ’633 patent identifies a (protected) packaging engine in a protection engine as a structure that causes mobile protection code to be communicated to an information-destination: “Packaging engine 403 provides for generating mobile protection code and protection policies, and for causing delivery thereof (typically with a detected-Downloadable) to a Downloadable-destination for protecting the Downloadable-destination against malicious operation attempts by the detected Downloadable.”

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

(Ex. 1001 at 12:38–43.)

92. The protection engine and its components, including the packing engine, can be hosted by a server or a firewall, distributed among several different servers or firewalls, or may be hosted on a standalone device. (*Id.* at 7:26–33.) The protection engine and its components, including the code detection engine, can be implemented in hardware, software, or a combination of hardware and software. (*Id.* at 7:66–8:2.)

93. Counsel informs me that the Blue Coat Court construed the function of “means for causing” to be “if the downloadable-information is determined to include executable code, causing mobile protection code to be communicated to at least one information-destination of the downloadable-information *without modifying the executable code.*” (Ex. 1036 at 8 (emphasis added).) I disagree with that interpretation because claim 13 is silent on whether or not the “means for causing” modifies executable code, and nothing in the ’633 patent indicates that the “means for causing” does not, in every embodiment, modify executable code. (*See* Ex. 1039 at 12–13; Ex. 1033 at 6–9.)

**D. The Priority Claims of the ’633 Patent**

94. I have looked at the various patents that purport to be related to the ’633 patent. These are found in Exhibits 1085, 1086, 1087, 1031, 1032, 1040, and 1083. I am also aware that Patent Owner has argued for an earlier priority date for

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

the '633 patent in *Ex Parte* Reexamination Control No. 90/013,016. (Ex. 1081 at 1-9).

95. I understand that the priority date for a particular claim is based in part on when, in a chain of related patents, the written description that supports that claim first appeared. For example, I have looked at the patents and applications listed above to determine which patent specifications disclose supporting material for claims 1-3, 8, 13-14, 19, 28, and 34 (which I refer to as “claimset 1”). The earliest disclosure that a POSA would recognize as providing a description of the subject matter of those claims was the '591 provisional application filed May 17, 2000. (Ex. 1040 at 4-9.) Specifically, the indicated subject matter in the following claims is missing from any priority applications before the '591 provisional application:

Claims 1-3: “transmitting from the computer mobile protection code to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code”

Claims 8 and 13: “causing mobile protection code (“MPC”) to be communicated . . . to at least one information-destination of the downloadable-information, if the downloadable-information is determined to include executable code”

Claim 14 and 19: “causing mobile protection code to be executed by the

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

mobile code executor at a downloadable-information destination such that one or more operations of the executable code at the destination, if attempted, will be processed by the mobile protection code”

Claim 28: “receiving a sandboxed package that includes mobile protection code (“MPC”) and a Downloadable and one or more protection policies at a computer at a Downloadable-destination”

Claim 34: “sandboxed package including a Downloadable and mobile protection code (“MPC”) for causing one or more Downloadable operations to be intercepted by the computer and for processing the intercepted operations by the computer, if the Downloadable attempts to initiate the operations.”

96. I also looked at the patents and applications listed above to determine which patent specifications disclose supporting material for claims 4, 6, and 7 (which I refer to as “claimset 2”). The earliest disclosure that a POSA would recognize as providing a description of the subject matter of those claims was the ’229 parent application filed on May 17, 2001. (Ex. 1094 at 32-33, 35-37, 45-47.) Specifically, the material in the following claims is missing from any priority applications before the ’229 application:

Claim 4: “determining comprises analyzing the downloadable-information for an included type indicator indicating an executable file type”

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

Claim 6: “determining comprises analyzing the downloadable-information for an included file type indictor and an information pattern corresponding to one or more information patterns that tend to be included within executable code”

Claim 7: “determining is conducted in accordance with the executable code characteristics”

## **V. OVERVIEW OF THE PRIOR ART**

### **A. Overview of Poison Java**

97. Poison Java is an article that was published in the August 1999 issue of *IEEE Spectrum* magazine. (Ex. 1004 at 1.) Poison Java discusses a security product developed by Trend Micro called Interscan AppletTrap, which is mentioned in the '591 provisional application.

98. Poison Java discloses an Internet-gateway based system for computer security. (Ex. 1004 at 5, col. 3.) The system described in Poison Java employs a proxy server that runs a series of “prefiltering” checks on downloaded Java applets. (*Id.*) First, the proxy server blocks unwanted applets by creating an MD5 hash code of the applet and comparing the MD5 hash code to a list of known, malicious applets. (*Id.*) Second, the proxy server runs certificate checks on the applet and blocks those that are unsigned, signed by an unrecognized certificate authority, or

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

signed with a signature that does not match the content. (*Id.*)

99. If it does not block an applet during prefiltering, the system described in Poison Java determines whether the applet will perform a potential malicious action, such as calling to system resources. (Ex. 1004 at 5, col. 3.) The proxy server then “wraps monitoring code around the applet and attaches the security policy.” (*Id.*) The applet, monitoring code, and policy are then delivered to the client. (*Id.* at 6, col. 1.) When the client initiates the applet, the monitoring code monitors the actions of the applet and compares them with the attached security policy. (*Id.*) If the actions violate the policy, the monitoring code responds with an action predetermined by the security administrator, such as blocking the applet and notifying the user. (*Id.*) Any applets determined to be malicious are added to the block list. (*Id.* at 6, col. 1-2).

100. As an expert in the field who was actively engaged in writing and researching the risks from applets, I regularly read publications by other researchers on the topic of java security. I remember reading a copy of Poison Java during the calendar year 1999 in the normal course of my activities as a professor/researcher. The fact that I received and read Poison Java during calendar year 1999 confirms that Poison Java was published and publicly available in that timeframe.



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

**B. Overview of Shin**

101. Shin discloses an HTTP proxy server that first determines whether downloaded data includes a Java applet (executable code). (Ex. 1009 at 17-18.) Specifically, Shin teaches scanning downloaded data for “<applet>” tags or a “magic byte sequence” that appears at the beginning of all Java class files. (*Id.*) If the proxy server determines that the data includes a Java applet, it inserts “safeguarding code” into the applet before passing the applet on to a client’s browser. (*Id.* at 2, 4-7.) The safeguarding code can be implemented as a class-level or method-level modification of the applet. (*Id.* at 4-7.) When an instrumented applet runs on a client computer, the safeguarding code can monitor and control resource usage and limit the functionality of the applet. (*Id.* at 4.)

102. As an expert in the field who was actively engaged in writing and researching the risks from applets, I regularly read publications by other researchers on the topic of java security. During the 1998 calendar year, I remember reading a copy of Shin in the normal course of my activities as a professor/researcher. The fact that I found and read Shin during calendar year 1998 confirms that Poison Java was published and publicly available in that timeframe.

**C. Overview of Brown**

103. Brown (Ex. 1041) is a guide to the Netscape Navigator 3 web browser, published in 1996, that describes Netscape Navigator 3’s functionality

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

and the execution of Java applets by the Navigator 3 browser.

**D. Poison Java, Shin, and Brown Are All Analogous Art**

104. I understand that to combine prior art references when evaluating validity, those references must generally be “analogous.” To be analogous, the references must be in the same field of endeavor as the ’633 patent, and/or must be pertinent to the problems to which the ’633 patent is directed. This requirement is met by the Poison Java, Shin, and Brown references.

105. Poison Java and Shin are both in the same field of endeavor as the ’633 patent, namely the field of computer security methods and systems, including content-scanning for program code. Poison Java and Shin are similarly directed to security programs that recognize potentially malicious code. (*See, e.g.*, Ex. 1004 at 5, col. 3; Ex. 1009 at 1-2.)

106. Poison Java and Shin are also analogous art to the ’633 patent because they utilize a similar protection method. Poison Java discloses an HTTP proxy server that wraps a potentially malicious downloadable with mobile code that monitors actions of the downloadable at the client. (Ex. 1004 at 5, col. 3, to 6, col. 1). Shin discloses an HTTP proxy server that inserts “safeguarding code” into Java applets in order to protect the client from certain kinds of hazardous run-time behaviors. (Ex. 1009 at 4, 14, 16). The applet and safeguarding code are then forwarded to the client computer. (*Id.* at 2.)

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

107. Brown is also analogous art to the '633 patent because it is highly pertinent to the problems addressed by the '633 patent; i.e., problems associated with downloading and running applets safely. (Ex. 1041 at 4-5, 6-13.) Brown would have served as a background reference to a POSA designing and implementing security programs for applets at the time of the effective filing date of the '633 patent.

## **VI. ANALYSIS**

### **A. Shin Renders Claims 1–4, 6–8, 13, 14, and 19 Obvious under 35 U.S.C. § 103(a)**

#### **1. Independent Claim 1**

##### **a. Claim element 1[b]: “receiving” limitation**

108. Shin discloses an “HTTP proxy server, written in Python, [that] performs forwarding of messages between client and web server, as well as transformation of applets.” (*See, e.g.*, Ex. 1009 at 14.) A POSA would have understood that, to forward the messages and applets (“downloadable-information”) from the source web server to the client, the proxy server in Shin must first receive them from the web server. (Ex. 1072 at 1, 3-4.)

##### **b. Claim element 1[c]: “determining” limitation**

109. Element 1[c] recites “determining, by the computer, whether the downloadable information includes executable code.” A Java applet is an internet-based program written in the Java programming language, which is made up of

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

executable code that can be downloaded and executed by any computer. (Ex. 1073 at 12, 16-17.) Shin discloses a firewall that identifies Java applets by scanning for <applet> tags, magic byte sequences, or file name extensions that are characteristic of Java applets. (Ex. 1009 at 17-18.)

110. Tags are used within HTML text to call out particular sections of the HTML. When downloaded HTML text includes a Java applet, the text will include an applet tag <applet> somewhere in the downloaded file. (Ex. 1074 at 1; Ex. 1019 at 1.) Therefore, when the firewall disclosed in Shin scans for <applet> tags, it is searching for a type indicator that a Java applet is included in the downloaded file.

111. In downloading or receiving information, firewalls and proxy servers receive an incoming stream of data bytes. (Ex. 1075 at 6, col. 1, “Firewall Design”; Ex. 1009 at 17-18.) The firewall disclosed in Shin scans this incoming data stream for a “magic byte sequence,” a specific sequence of bytes that is known to appear in Java applets. (Ex. 1009 at 17-18.) Therefore, when Shin’s firewall scans for this known “magic byte sequence,” it is scanning the downloadable-information to determine whether it includes a Java applet or executable code.

112. A computer file name typically includes an extension corresponding to its file type. For example, .TXT indicates a text file, and .DOC or .DOCX indicates a Microsoft Word file. (Ex. 1076 at 1-7.) There are also file extensions or type indicators that are characteristic of Java applets, such as .CLASS and .JAR.

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

(Ex. 1077 at 5.) Therefore, when the firewall disclosed in Shin scans downloadable-information for file-name extensions such as “.class” (Ex. 1009 at 17), it is scanning for a type indicator that indicates executable code—specifically, a Java applet.

113. Because the firewall disclosed in Shin scans for <applet> tags, the “magic byte sequence,” or file extensions associated with Java applets, Shin teaches a firewall capable of scanning an incoming information stream and determining, based on a file’s executable code characteristics, whether it includes a Java applet—executable code.

114. Shin discloses at least one of the same techniques for detecting executable code that are disclosed in the ’633 patent, namely looking for “a Java class header for Java applets,” which is the “magic byte sequence” discussed above. (Ex. 1075 at 12; Ex. 1001 at 14:60-65 (“File-reader 502 can, for example, be configured to analyze a received potential-Downloadable for a file header . . . such as . . . *a Java class header for Java applets.*”) (emphasis added).)

## **2. Claim 3**

115. The HTTP proxy server in Shin falls within the BRI of the recited “information re-communicator.” A POSA would have understood that an HTTP proxy server is a type of “network server” because proxy servers are widely understood as being deployed in computer networks, as evidenced by, e.g., the

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

1994 paper “World-Wide Web Proxies” by Luotonen *et al.* (Ex. 1072 at 2-6.)

### **3. Claims 6 and 7**

116. As discussed above in connection with Element [1c], the “magic byte sequence” for which the firewall in Shin scans is a Java class header that is required to appear near the beginning of every Java class file. (Ex. 1075 at 12.) Therefore, the “magic byte sequence” discussed in Shin is an “information pattern” that “tend[s] to be included within executable code.” Similarly, the applet tags, magic byte sequence (Java class header), and “.class” filename extension that Shin analyzes to detect Java applets (executable code) are all “executable code characteristics,” as recited in dependent claim 7. (*See, e.g.*, Ex. 1075 at 11-13.)

117. Regarding checking for both a file type indicator (e.g., a filename extension such as “.class”) *and* an information pattern like the Java class header, as recited in claim 6, it would have been obvious to a POSA, in light of the well-known “Defense in Depth” concept discussed above in the Technology Tutorial, to employ both techniques together to reduce the likelihood of error in identifying executable objects. For example, Martin teaches that it is better to combine <applet> tag scanning (scanning for a file type indicator) with scanning for the Java class header “CA FE BA BE” (scanning for an information pattern) because the latter can serve as back up, if the <applet> tag makes it through the first line of defense. (Ex. 1075 at 12, col. 2.) Such a combined approach would have involved

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

the mere combining of prior-art elements according to known methods to yield predictable results and, furthermore, would have been obvious to a POSA to try. (MPEP § 2143(I), Rationales (A) and (E).) Analyzing a data stream for specified filename extensions and analyzing the data stream for a particular byte sequence (e.g., the Java class header) both involve comparing the incoming data with predetermined information patterns.

#### 4. Claim 8

118. The specification of the '633 patent discloses that the “information monitor,” “content inspection engine,” and “protection agent engine” recited in claim 8 can be implemented in software and/or hardware. (Ex. 1001 at 7:65–8:63.) A POSA would have understood that Shin’s proxy server and firewall, which covers the functionality of those three elements, can likewise be implemented in software and/or hardware. For example, Peterson’s 1999 paper discusses the software vs. hardware implementation tradeoffs in the design of a general-purpose network router. (Ex. 1078 at 2.) A POSA would also have understood that Shin’s proxy server and/or firewall implementation of the recited “information monitor,” “content inspection engine,” and “protection agent engine” could be realized as separate software modules or as one or more combined software modules. In fact, implementing separate modules to perform the functions recited in claim 8 was an obvious and commonplace design choice, as evidenced by Wieringa’s 1998 paper

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

on traceability and modularity in software design. (Ex. 1079 at 1-6.)

### 5. Claim 13

119. Shin renders obvious the “means for determining” recited in claim 13 because it discloses both the function (see my analysis of Element [1c] above) and the structure associated with this term. The BRI of “means for determining” is “a code detection engine,” which the ’633 patent discloses can be implemented as one or more software modules that are hosted by a server or a firewall or that are distributed among a server and a firewall. (Ex. 1001 at 7:65–8:2, 8:47-67.) To a POSA, it would have been an obvious design choice to implement the “determining” function performed by Shin’s firewall as a software module executing on the firewall, particularly since Shin expressly teaches that very kind of embodiment in connection with its Java-bytecode-modification system, which includes an HTTP proxy server written in the Python language running on a general-purpose computer. (Ex. 1009 at 14.)

120. Similar reasoning applies to the “means for causing” recited in claim 13. The BRI of “means for causing” is “a packaging engine,” which the ’633 patent discloses can be implemented as one or more software modules that are hosted by a server or a firewall or that are distributed among a server and firewall. (Ex. 1001 at 7:65–8:2, 8:47-67.) As with the “means for determining,” it would have been an obvious design choice to implement the “causing” function



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

performed by Shin’s server as a software module executing on the HTTP proxy server because that’s how Shin implemented it. (Ex. 1009 at 14 (“Our HTTP proxy server, written in Python, performs forwarding of messages between client and web server . . . . [O]ur proxy server [runs on] a Sun Ultra Enterprise 3000 which has two 248MHZ Ultrasparc processors”).)

**6. Independent Claim 14**

**a. Claim element 14[a]: “computer program product”**

121. Shin discloses an HTTP proxy server, written in the Python programming language, that is directed to “computer security.” (Ex. 1009 at 1-2, 14-15.) Shin also discloses an HTTP client implemented as a Java program. (Ex. 1009 at 14.) A POSA would readily have understood that the Python and Java code implementing, respectively, the HTTP proxy server and the HTTP client would be stored on a “computer usable medium,” such as a magnetic data storage medium, as evidenced by, e.g., a U.S. patent to Ulrich *et al.* (Ex. 1080 at 1:18-19 (“Computer systems commonly utilize hard disc drives as a nonvolatile way to store data”).)

**b. Claim element 14[c]: “information re-communicator” and “mobile code executor”**

122. A POSA would have understood that the Java Virtual Machine disclosed in Shin (Ex. 1009 at 1-2, 4) would be executed within the HTTP client (Ex. 1009 at 14-15) disclosed in Shin. (Ex. 1073 at 12, 16-17.) Shin itself would

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

have motivated a POSA to use the Java Virtual Machine at the HTTP client (e.g., in the web browser running on the client computer) in order to execute the applets provided by the HTTP proxy server: “The *HTTP client is a Java program* which sends a request to a web server, receives its reply from the server, and measures the time it takes to receive the reply.” (Ex. 1009 at 14 (emphasis added).) A “Java program” requires a Java Virtual Machine for execution. (Ex. 1073 at 57-68.) Therefore, Shin teaches, or at least suggests, executing a Java Virtual Machine at the HTTP client.

**c. Claim element 14[d]: “receiving” limitation**

123. Shin discloses an HTTP proxy server that receives Java applets, modifies them with “safeguarding code,” and forwards them to a client computer running a web browser. (Ex. 1009 at 2, 4.) As discussed above in connection with Element 1[c], a Java applet is an internet-based program written in the Java programming language with its associated classes, which is made up of executable code that can be downloaded and executed by any computer. (Ex. 1073 at 12, 16-17.) Therefore, the HTTP proxy server in Shin receives downloadable-information that includes executable code.

**d. Claim element 14[e]: “causing” limitation**

124. As discussed above in connection with Element 14[c], A POSA would have understood that the Java Virtual Machine disclosed in Shin (Ex. 1009 at 1-2,

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

4) would be executed within the HTTP client (Ex. 1009 at 14-15) disclosed in Shin. (Ex. 1073 at 12, 16-17.) Shin itself would have motivated a POSA to use the Java Virtual Machine at the HTTP client (e.g., in the web browser running on the client computer) in order to execute the applets provided by the HTTP proxy server: “The *HTTP client* is a *Java program* which sends a request to a web server, receives its reply from the server, and measures the time it takes to receive the reply.” (Ex. 1009 at 14 (emphasis added).) A “Java program” requires a Java Virtual Machine for execution. (Ex. 1073 at 57-68.) Therefore, Shin teaches, or at least suggests, executing a Java Virtual Machine at the HTTP client.

**B. Poison Java in view of Shin Renders Claim 1 Obvious under 35 U.S.C. § 103(a)**

**1. Independent Claim 1**

**a. Claim element 1[c]: “determining” limitation**

125. According to Poison Java, a prerequisite for AppletTrap’s successful operation is the ability to identify Java applets in an incoming data stream that includes both executable and non-executable objects. That is, AppletTrap must be able to “weed[] out unwanted applets as HTML pages are downloaded.” (Ex. 1004 at 5, col. 3.) Poison Java does not discuss specific techniques for identifying executable code at the proxy server, but Shin expressly discloses several such techniques, which I have discussed above in connection with my analysis of Element 1[c] relative to the Shin reference. (*See* Ex. 1009 at 17-18.) A POSA

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

would have understood that the system in Poison Java must include some mechanism for identifying Java applets, or it would be inoperable. However, Poison Java does not expressly disclose this.

126. Because of the similarity of the solutions in Poison Java and Shin (proxy server, instrumentation with protective code, etc.) and the need, in Poison Java, to identify Java applets for potential wrapping in monitoring code, it would have been obvious to a POSA, using known methods of coding, to deploy, in the system described in Poison Java, the applet filtering techniques disclosed in Shin. Such a combination would require no modification of the teachings in Poison Java because the identification of executable code, as I mentioned above, is a prerequisite step to the rest of what Poison Java describes. A POSA could, thus, test and refine the executable-code-detection techniques independently of the rest of the AppletTrap system described in Poison Java so that the predictability and likelihood of success of the eventual combination would be enhanced. Therefore, such a combination would involve the combining of prior art elements according to known methods to yield predictable results. (MPEP § 2143(I), Rationale (A).)

127. Furthermore, adding Shin's applet filtering techniques to the system described in Poison Java would have been "obvious to try" for a POSA because it would involve choosing from among a finite number of identified, predictable solutions with a reasonable probability of success. (MPEP § 2143(I), Rationale

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

(E.) Shin identifies a finite number of identified, predictable potential solutions to the problem of identifying and filtering potentially malicious applets, including (1) checking downloaded files for <applet> tags; (2) checking downloaded files for “.class”; (3) checking downloaded files for a “magic byte sequence”; and (4) authenticating digital signatures. (Ex. 1009 at 17-18.) A POSA would have had a reasonable expectation of success in combining Shin’s applet filtering techniques with the system described in Poison Java, particularly since Shin already describes implementing these methods at both a firewall and a client web browser. (Ex. 1009 at 17-18.)

**C. Poison Java in view of Brown Renders Claims 14, 19, and 34 Obvious under 35 U.S.C. § 103(a)**

**1. Independent Claim 14**

**a. Claim element 14[a]: “computer program product”**

128. Poison Java discloses a computer security program called AppletTrap written in computer-readable program code: “A hybrid solution to supplementing Java security was recently released by the author’s company, Trend Microsystems Inc. Called InterScan AppletTrap, the software integrates elements of both client- and server-based solutions [Fig. 4].” (Ex. 1004 at 5, col. 3.) A POSA would readily have understood that the AppletTrap program code implementing the proxy server would be stored on a “computer usable medium,” such as a magnetic data storage medium, as evidenced by, e.g., a U.S. patent to Ulrich *et al.* (Ex. 1080 at 1:18-19

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

(“Computer systems commonly utilize hard disc drives as a nonvolatile way to store data”).)

**b. Claim element 14[c]: “information re-communicator” and “mobile code executor”**

129. The Web browser in Poison Java corresponds to the recited “mobile code executor.” Poison Java states, “The HTML page, along with the instrumented applets, is then delivered to the client and displayed on its Web browser.” (Ex. 1004 at 6, col. 1.) It would have been obvious to a POSA that the instrumented applets transmitted to the client computer are executed by the Web browser in Poison Java, as taught by Brown. (*See* Ex. 1041 at 4-5, 6-13.) In fact, Poison Java itself suggests that the instrumented applets are run in the Web browser. The very next sentence in Poison Java following the one just quoted above states, “*Once the applet is invoked*, AppletTrap’s monitoring code extracts information about the resources that will be used by the applet and ascertains the permissibility of this action by comparing it with the attached security policy.” (Ex. 1004 at 6, col. 1.) That statement, immediately following the above-quoted sentence describing what the Web browser receives and displays, implies that the Web browser also “invokes” (executes) the applet. This is consistent with how Java applets normally work—namely, that they are downloaded from a network server to a client computer and executed by a Java Virtual Machine associated with a Web browser.

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

(Ex. 1073 at 12, 16-17, 56-67.)

**c. Claim element 14[d]: “receiving” limitation**

130. Poison Java discloses a proxy server that receives and “weeds out” (prefilters) unwanted applets as HTML pages are being downloaded and wraps monitoring code around applets that pass the prefiltering tests before forwarding them to a client computer. (Ex. 1004 at 5, col. 3, to 6, col. 1.) As I discussed above, a Java applet is an internet-based program written in the Java programming language with its associated classes, which is made up of executable code that can be downloaded and executed by any computer. (Ex. 1073 at 12, 16-17.) Therefore, Java applets are a type of executable program code.

**d. Claim element 14[e]: “causing” limitation**

131. The proxy server in Poison Java instruments (wraps monitoring code around) Java applets that pass a prefiltering step. (Ex. 1004 at 5, col. 3.) As discussed above, the Web browser in Poison Java corresponds to the recited “mobile code executor” that executes the instrumented applets. Poison Java states, “The HTML page, along with the instrumented applets, is then delivered to the client and displayed on its Web browser.” (Ex. 1004 at 6, col. 1.) It would have been obvious to a POSA that the instrumented applets, including the monitoring code (the recited “mobile protection code”) added by the proxy server, are executed by the Web browser in Poison Java, as taught by Brown. (*See* Ex. 1041 at

Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

4-5, 6-13.) In fact, Poison Java itself suggests that the instrumented applets and associated monitoring code are run in the Web browser. The very next sentence in Poison Java following the one just quoted above states, “*Once the applet is invoked*, AppletTrap’s monitoring code extracts information about the resources that will be used by the applet and ascertains the permissibility of this action by comparing it with the attached security policy.” (Ex. 1004 at 6, col. 1.) That statement, immediately following the above-quoted sentence describing what the Web browser receives and displays, implies that the Web browser also “invokes” (executes) the applet. This is consistent with how Java applets normally work—namely, that they are downloaded from a network server to a client computer and executed by a Java Virtual Machine associated with a Web browser. (Ex. 1073 at 57-68.)

## **2. Independent Claim 34**

### **a. Claim element 34[b]: “mobile code executor”**

132. As discussed above, the Web browser in Poison Java corresponds to the recited “mobile code executor.” Poison Java states, “The HTML page, along with the instrumented applets, is then delivered to the client and displayed on its Web browser.” (Ex. 1004 at 6, col. 1.) It would have been obvious to a POSA that the instrumented applets, including the monitoring code (the recited “mobile protection code”) added by the proxy server, are executed by the Web browser in



Declaration of Aviel D. Rubin

Petition for *Inter Partes* Review of Patent No. 7,647,633

Poison Java, as taught by Brown. (*See* Ex. 1041 at 4-5, 6-13.) In fact, Poison Java itself suggests that the instrumented applets and associated monitoring code are run in the Web browser. The very next sentence in Poison Java following the one just quoted above states, “*Once the applet is invoked, AppletTrap’s monitoring code extracts information about the resources that will be used by the applet and ascertains the permissibility of this action by comparing it with the attached security policy.*” (Ex. 1004 at 6, col. 1.) That statement, immediately following the above-quoted sentence describing what the Web browser receives and displays, implies that the Web browser also “invokes” (executes) the applet. This is consistent with how Java applets normally work—namely, that they are downloaded from a network server to a client computer and executed by a Java Virtual Machine associated with a Web browser. (Ex. 1073 at 57-68.)

## **VII. SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS**

133. I have reviewed the alleged evidence of secondary considerations of non-obviousness that Patent Owner has presented in *Ex Parte* Reexamination No. 90/013,016. (*See* Ex. 1042 at 30-34; Ex. 1088 at 1-5.) Nothing in that alleged evidence has altered my opinion that the Petitioned Claims of the ’633 patent are invalid over the prior-art references I have reviewed and analyzed above. In particular, I do not see any nexus between the Petitioned Claims and Patent Owner’s alleged evidence of non-obviousness.

Declaration of Aviel D. Rubin  
Petition for *Inter Partes* Review of Patent No. 7,647,633

### VIII. CONCLUSION

134. I reserve the right to offer opinions relevant to the invalidity of the '633 patent claims at issue and/or offer testimony in support of the Declaration.

135. In signing this Declaration, I recognize that the Declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case. If required, I will appear for cross-examination at the appropriate time.

136. I hereby 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 further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. § 1001.

Dated: September 24, 2015

Respectfully submitted,



---

Aviel D. Rubin

# **EXHIBIT 9**

1 PAUL J. ANDRE (State Bar No. 196585)  
2 pandre@kramerlevin.com  
3 LISA KOBIALKA (State Bar No. 191404)  
4 lkobialka@kramerlevin.com  
5 JAMES HANNAH (State Bar No. 237978)  
6 jhannah@kramerlevin.com  
7 KRAMER LEVIN NAFTALIS & FRANKEL  
8 LLP  
9 990 Marsh Road  
10 Menlo Park, CA 94025  
11 Telephone: (650) 752-1700  
12 Facsimile: (650) 752-1800  
13 *Attorneys for Plaintiff*  
14 FINJAN, INC.

QUINN EMANUEL URQUHART &  
SULLIVAN, LLP  
Sean Pak (Bar No. 219032)  
seanpak@quinnemanuel.com  
50 California Street, 22<sup>nd</sup> Floor  
San Francisco, California 94111-4788  
Telephone: (415) 875-6600  
Facsimile: (415) 875-6700

David A. Nelson (*pro hac vice*)  
davenelson@quinnemanuel.com  
500 W. Madison Street, Suite 2450  
Chicago, Illinois 60661-2510  
Telephone: (312) 705-7400  
Facsimile: (312) 705-7401

Alexander Rudis (*pro hac vice*)  
alexanderrudis@quinnemanuel.com  
51 Madison Avenue, 22nd Floor  
New York, NY 10010  
Telephone: (212) 849-7000  
Facsimile: (212) 849-7100

*Attorneys for Defendant* SYMANTEC  
CORPORATION

17 **IN THE UNITED STATES DISTRICT COURT**  
18 **FOR THE NORTHERN DISTRICT OF CALIFORNIA**  
19 **SAN FRANCISCO DIVISION**

20 FINJAN, INC., a Delaware Corporation,  
21 Plaintiff,  
22 v.  
23 SYMANTEC CORP., a Delaware Corporation,  
24 Defendant.

Case No.: 3:14-CV-02998-HSG

**JOINT CLAIM CONSTRUCTION AND  
PRE-HEARING STATEMENT  
PURSUANT TO PATENT LOCAL RULE  
4-3**

Pursuant to the Court’s Scheduling Order (Dkt. No. 56) and Patent L.R. 4-3 Plaintiff Finjan, Inc. (“Finjan”) and Defendant Symantec Corporation (“Symantec” or “Defendant”) hereby submit this Joint Claim Construction and Pre-Hearing Statement.

**I. PATENT L.R. 4-3(a): CLAIM TERM(S) ON WHICH THE PARTIES AGREE.**

During the meet and confer process, the parties agreed to the construction of the following term:

Claim Term	Agreed Construction
Downloadable security profile that identifies suspicious code in the received Downloadable	a profile that identifies code in the received Downloadable that performs hostile or potentially hostile operations

**II. PATENT L.R. 4-3(b): PROPOSED CONSTRUCTION OF EACH DISPUTED TERM.**

The parties’ proposed claim constructions are provided below.<sup>1</sup> All supporting evidence for the parties’ claim constructions is provided in Exhibit A. The parties reserve their rights to cite additional supporting evidence based on arguments raised in the claim construction briefs.

U.S. Patent No. 6,154,844			
Claim Term	Claim(s)	Finjan’s Proposed Construction	Defendants’ Proposed Construction
Downloadable	1, 15, 41, 43	an executable application program, which is downloaded from a source computer and run on the destination computer	mobile code that is requested by an ongoing process and downloaded from a source computer to a destination computer for automatic execution
means for receiving a Downloadable	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> receiving a Downloadable	means-plus-function under § 112, ¶ 6  <b>Function:</b> receiving a Downloadable  <b>Corresponding structure:</b>

<sup>1</sup> The identified claim numbers refer to the asserted independent claims where the term appears unless the term only appears in certain asserted dependent claims which are identified.

U.S. Patent No. 6,154,844			
Claim Term	Claim(s)	Finjan's Proposed Construction	Defendants' Proposed Construction
		<b>Structure:</b> Downloadable file interceptor	indefinite for failure to disclose corresponding structure/algorithm
means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable	43	Governed by 35 U.S.C. § 112(6): <b>Function:</b> generating a first Downloadable security profile that identifies suspicious code in the received Downloadable  <b>Structure:</b> content inspection engine	means-plus-function under § 112, ¶ 6  <b>Function:</b> generating a first Downloadable security profile that identifies suspicious code in the received Downloadable  <b>Corresponding structure:</b> a processor programmed to perform the algorithm disclosed at col. 5, lines 42-45 and col. 9, lines 20-42 of U.S. Patent No. 6,092,194
means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Structure:</b> content inspection engine	means-plus-function under § 112, ¶ 6  <b>Function:</b> linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Corresponding structure:</b> a processor programmed to perform the algorithm of steps 630 and 645 disclosed at Fig. 6, col. 8 lines 65-67, col. 6, lines 13-24, and col. 5, lines 3-13
before a web server makes the Downloadable available to web clients	1, 15, 41, 43	No construction necessary – Plain and ordinary meaning.	before [a/the] non-network gateway web server make[s] the Downloadable available to web clients
a first content inspection engine	15	No construction necessary – Plain and ordinary meaning.	means-plus-function under § 112, ¶ 6

U.S. Patent No. 6,154,844			
Claim Term	Claim(s)	Finjan's Proposed Construction	Defendants' Proposed Construction
for using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients			<p><b>Function:</b> using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients</p> <p><b>Corresponding structure:</b> a processor programmed to perform the algorithm disclosed at col. 5, lines 42-45 and col. 9, lines 20-42 of U.S. Patent No. 6,092,194 , and the algorithm of steps 630 and 645 disclosed at Fig. 6, col. 8 lines 65-67, col. 6, lines 13-24, and col. 5, lines 3-13 of the '844 patent.</p>

U.S. Patent No. 7,613,926			
Claim Term	Claim(s)	Finjan's Proposed Construction	Defendants' Proposed Construction
Downloadable	1, 8, 15, 22, 29, 30	an executable application program, which is downloaded from a source computer and run on the destination computer	mobile code that is requested by an ongoing process and downloaded from a source computer to a destination computer for automatic execution
append[er/ed/ing]	1,8, 29	No construction necessary – Plain and ordinary meaning.	attach to the end of
database	1, 8, 15, 22, 29, 30	a collection of interrelated data organized according to a database schema to serve one or more applications	organized collection of data

**U.S. Patent No. 8,667,494**

<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan's Proposed Construction</b>	<b>Defendants' Proposed Construction</b>
Downloadable	1, 10	an executable application program, which is downloaded from a source computer and run on the destination computer	mobile code that is requested by an ongoing process and downloaded from a source computer to a destination computer for automatic execution
database	1, 10	a collection of interrelated data organized according to a database schema to serve one or more applications	organized collection of data

**U.S. Patent No. 7,756,996**

<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan's Proposed Construction</b>	<b>Defendants' Proposed Construction</b>
network gateway computer	1	No construction necessary – Plain and ordinary meaning.	a computer that is a point of contact between different networks
non-HTTP management data	1, 4, 7	No construction necessary – Plain and ordinary meaning.	management data that the management server transmits and receives using a non-HTTP transport protocol

**U.S. Patent No. 7,757,289**

<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan's Proposed Construction</b>	<b>Defendants' Proposed Construction</b>
protecting a computer from dynamically generated malicious content	1, 10, 19, 22, 35, 41	No construction necessary of preamble. If construed, plain and ordinary meaning should apply.	Preamble is limiting.  protecting a computer from malicious content that is generated at run-time
content processor	10	No construction necessary – Plain and ordinary meaning.	software that renders the content for interactive viewing on a display monitor



**U.S. Patent No. 7,757,289**

<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction</b>	<b>Defendants’ Proposed Construction</b>
said content processor (i) suspends processing of the modified content after said client transmitter transmits the input to said security computer, and (ii) resumes processing of the modified content after said client receiver receives the indicator from said security computer.	16	Not indefinite– plain and ordinary meaning.  To the extent a construction is required, the plain and ordinary meaning of this terms is:  during processing of the modified content, said content processor first suspends processing of the modified content after said client transmitter transmits the input to said security computer, and then resumes processing of the modified content after said client receiver receives the indicator from said security computer.	indefinite
inspection[s]	1, 10, 19, 22, 35, 41	No construction necessary – Plain and ordinary meaning	scanning for the presence of potentially malicious operations

**U.S. Patent No. 8,141,154**

<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction</b>	<b>Defendants’ Proposed Construction</b>
protecting a computer from dynamically generated malicious content	1, 6	No construction necessary of preamble. If construed, plain and ordinary meaning should apply.	Preamble is limiting.  protecting a computer from malicious content that is generated at run-time
content processor	1, 6	No construction necessary – Plain and ordinary meaning.	software that renders the content for interactive viewing on a display monitor
a call to a first function	1, 4, 6, 10	No construction necessary – Plain and ordinary	a call to a function different from the second function

<b>U.S. Patent No. 8,141,154</b>			
<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan's Proposed Construction</b>	<b>Defendants' Proposed Construction</b>
		meaning.	
said content processor (i) suspends processing of the content after said transmitter transmits the input to the security computer, and (ii) resumes processing of the content after said receiver receives the [indicator/modified input variable]	2, 7	Not indefinite– plain and ordinary meaning.  To the extent a construction is required, the plain and ordinary meaning of this terms is:  during processing content received over a network, said content processor first suspends processing of the content after said transmitter transmits the input to the security computer, and then resumes processing of the content after said receiver receives the [indicator/modified input variable]	indefinite
[invoking / invoke / calling] a second function	1, 4, 6, 10	No construction necessary – Plain and ordinary meaning.	[invoking / invoke / calling] a function different from the first function
inspection	1, 4, 6, 10	No construction necessary – Plain and ordinary meaning.	Scanning for the presence of potentially malicious operations

<b>U.S. Patent No. 8,015,182</b>			
<b>Claim Term</b>	<b>Claim(s)</b>	<b>Finjan's Proposed Construction</b>	<b>Defendants' Proposed Construction</b>
dynamically updating the presentation when additional security assessments are received	1, 8, 15	No construction necessary – Plain and ordinary meaning.	updating the presentation of the search results summary and a portion of the security assessments to present additional security assessments when the additional security assessments are received

U.S. Patent No. 7,930,299			
Claim Term	Claim(s)	Finjan’s Proposed Construction	Defendants’ Proposed Construction
dynamically updat[e/es/ing] the combined search and security results summary	1, 13, 20	No construction necessary – Plain and ordinary meaning.	updating the presented search results and assessments of potential security risks to present additional assessments of potential security risks after additional assessments of potential security risks are received

**III. PATENT L.R. 4-3(c): IDENTIFICATION OF MOST SIGNIFICANT TERMS.**

FINJAN’S STATEMENT:

Finjan does not consider any of the disputed terms significant or case or claim dispositive. At this stage of the litigation, it is unclear whether any of the constructions will be case or claim dispositive, particularly due to the deficiencies in Symantec’s invalidity contentions, which are vague, ambiguous, and fail to particularly point out Symantec’s invalidity theories with respect to the asserted patents.

DEFENDANT’S STATEMENT:

Symantec considers at least the following ten (10) terms to be significant to the resolution of the case<sup>2</sup>.

No.	Patent(s)	Term
1.	'289/'154	protecting a computer from dynamically generated malicious content
2.	'289/'154	content processor

<sup>2</sup> Finjan objects to Symantec’s third identified claim term as actually being two separate terms: “dynamically updating the presentation when additional security assessments are received” (‘182 Patent) and “dynamically updat[e/es/ing] the combined search and security results summary” (‘299 Patent). Symantec believes that “dynamically updat[e/es/ing] the combined search and security results summary” and “dynamically updating the presentation when additional security assessments are received” in the ‘299 and ‘182 patents are from related patents and are very similar in substance. Symantec proposes very similar constructions for both, relies on the same evidence to support its constructions, and intends to brief them together, and therefore, it is Symantec’s position that these phrases constitute a single term.

3.	'299/'182	dynamically updat[e/es/ing] the combined search and security results summary / dynamically updating the presentation when additional security assessments are received
4.	'996	non-HTTP management data
5.	'289/'154	inspection[s]
6.	'926/'494	database
7.	'844/'926/'494	Downloadable
8.	'844	means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable
9.	'844	means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients
10.	'996	network gateway computer

At this stage of the litigation, it is unclear whether any of the constructions will be case or claim dispositive, particularly due to the deficiencies in Finjan's infringement contentions, which are vague, ambiguous, and fail to particularly point out Finjan's infringement theories with respect to the 135 patent claims that it has asserted.

THE PARTIES' IDENTIFICATION OF 10 CLAIM TERMS FOR CLAIM CONSTRUCTION

BRIEFING:

Finjan identifies the following 5 terms for briefing:

No.	Patent(s)	Term
1.	'844/'926/'494	Downloadable
2.	'844	means for receiving a Downloadable
3.	'844	means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable
4.	'844	means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients
5.	'926/'494	database

Symantec identifies the following 5 terms for briefing<sup>3</sup>:

No.	Patent(s)	Term
1.	'289/'154	protecting a computer from dynamically generated malicious content
2.	'289/'154	content processor
3.	'299/'182	dynamically updat[e/es/ing] the combined search and security results summary / dynamically updating the presentation when additional security assessments are received
4.	'996	non-HTTP management data
5.	'289/'154	inspection[s]

**IV. PATENT L.R. 4-3(d): TIME FOR CLAIM CONSTRUCTION HEARING.**

The parties anticipate that they will not require more than 4 hours for the entire claim construction hearing.

**V. PATENT L.R. 4-3(e): WITNESSES AT CLAIM CONSTRUCTION HEARING.**

**Finjan's Statement:**

Finjan intends to offer a declaration and may present live witness testimony from Dr. Nenad Medvidovic, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089, regarding a tutorial of the relevant technology, the technical background of the asserted patents, the qualifications of one of skill in the art at the time of the inventions, how the above terms are understood by one of skill in the art, and to support Finjan's claim construction positions. Furthermore, Dr. Nenad Medvidovic will offer an opinion regarding the definiteness of the claims and rebut any testimony or opinions offered by Defendant's expert witness, Dr. Ford.

**Defendant's Statement:**

Symantec intends to offer a declaration and may present live witness testimony from Dr. Richard Ford, Dept. of Computer Sciences, Florida Institute of Technology, 150 W. University Blvd., Melbourne, FL 32901 regarding a tutorial of the relevant technology, the technical background of the asserted patents, the qualifications of a person of ordinary skill in the art at the time of the alleged

<sup>3</sup> See footnote 2.

1 inventions, how the terms would be understood by a person of ordinary skill at the time of the alleged  
2 inventions, and why Symantec's construction of the disputed terms is proper, Furthermore, Dr. Ford  
3 may offer an opinion regarding indefiniteness of the claims and will rebut testimony or opinions  
4 offered by Dr. Medvidovic.

5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

Respectfully submitted,

Dated: March 16, 2015

By: /s/ Paul J. Andre  
Paul J. Andre  
Lisa Kobialka  
James Hannah  
KRAMER LEVIN NAFTALIS  
& FRANKEL LLP  
990 Marsh Road  
Menlo Park, CA 94025  
Telephone: (650) 752-1700  
Facsimile: (650) 752-1800  
pandre@kramerlevin.com  
lkobialka@kramerlevin.com  
jhannah@kramerlevin.com

*Attorneys for Plaintiff*  
FINJAN, INC.

Dated: March 16, 2015

By: /s/ Alexander Rudis  
QUINN EMANUEL URQUHART &  
SULLIVAN, LLP  
Sean Pak (Bar No. 219032)  
seanpak@quinnemanuel.com  
50 California Street, 22<sup>nd</sup> Floor  
San Francisco, California 94111-4788  
Telephone: (415) 875-6600  
Facsimile: (415) 875-6700

David A. Nelson (*pro hac vice*)  
davenelson@quinnemanuel.com  
500 W. Madison Street, Suite 2450  
Chicago, Illinois 60661-2510  
Telephone: (312) 705-7400  
Facsimile: (312) 705-7401

Alexander Rudis (*pro hac vice*)  
alexanderrudis@quinnemanuel.com  
51 Madison Avenue, 22<sup>nd</sup> Floor  
New York, NY 10010  
Telephone: (212) 849-7000  
Facsimile: (212) 849-7100

*Attorneys for Defendant* SYMANTEC  
CORPORATION

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28

In accordance with Civil Local Rule 5-1(i)(3), I attest that concurrence in the filing of this document has been obtained from any other signatory to this document.

/s/ Paul J. Andre  
Paul J. Andre



## **EXHIBIT A**

**EXHIBIT A**

<b>U.S. PATENT NO. 6,154,844</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
1.	Downloadable	1, 15, 41, 43	<p>an executable application program, which is downloaded from a source computer and run on the destination computer</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs. 1-8;                      Claims 1, 5-8, 11, 15-17, 22-23, 32, 41-44;                      Col. 1, ll. 23-27, 37-59, 62-67;                      Col. 2, ll. 1-67;                      Col. 3, ll. 1-7; 66-67;                      Col. 4, ll. 1-64;                      Col. 5, ll. 63-67;                      Col. 6, ll. 1-24;                      Col. 8, ll. 37-67;                      Col. 9, ll. 1-18; 19-67;                      Col. 10, ll. 1- 24, 66-67; and                      Col. 11, ll. 1-11.</p> <p>’844 Patent File History including: September 2, 1999 Non-Final Rejection; November 23, 1999 Response to Non-Final Office Action; February 8, 2000 Non-Final Rejection; May 16, 2000 Response to Non-Final Action; and July 13, 2000 Notice of Allowance.</p>	<p>mobile code that is requested by an ongoing process and downloaded from a source computer to a destination computer for automatic execution</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’844 patent at Abstract, 1:37-59, 1:62-2:2.</p> <p>U.S. Patent No. 6,092,194 file history, 10/27/1999 Preliminary Amendment at 6.</p> <p>U.S. Patent No. 6,092,194 file history, 1/3/2000 Notice of Allowance.</p> <p>Provisional Application No. 60/030,639 at 1-2, Appendix.</p> <p>U.S. Patent No. 6,804,780 file history, 7/31/2003 Amendment and Response to Office Action at 7.</p> <p>U.S. Patent No. 6,167,520 at 1:24-38.</p> <p>U.S. Patent No. 7,613,926 at 6:3-18.</p>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p><b><u>Extrinsic Evidence</u></b></p> <p>Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper construction of the term from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p> <p>February 29, 2012 Order Construing the Terms of U.S. Patent Nos. 6,092,194 &amp; 6,480,962 – Finjan Inc. v. McAfee, Inc. et al. C.A. No. 10-cv-593 (GMS).</p> <p>August 12, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Websense, Inc., Civ. No. 13-cv-04398-BLF.</p> <p>October 14, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Sophos Inc., Civ. No. 14-cv-01197-WHO.</p> <p>October 20, 2014 Claim Construction Order – Finjan Inc. v. Blue Coat Systems, Inc., Civ. No.</p>	<p><b><u>Extrinsic Evidence:</u></b></p> <p>Finjan’s Opening Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 112).</p> <p>Finjan’s Answering Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 125).</p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “Downloadable”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’844 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the term “Downloadable” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</li> <li>4) why Symantec’s proposed construction for this term is proper.</li> </ol>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
			5:13-cv-03999-BLF.  January 26, 2015, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Proofpoint, Inc. et al., Civ. No. 13-cv-05808-BLF.  The intrinsic and extrinsic evidence cited by Defendant.	
2.	means for receiving a Downloadable	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> receiving a Downloadable  <b>Structure:</b> Downloadable file interceptor  <u><b>Intrinsic Evidence</b></u> Abstract; Figs. 1-8; Claims 10, 13, 15, 18, 19, 20, 21, 28, 36, 37, 43; Col. 2, ll. 3-60; Col. 4, ll. 59-64; Col. 5, ll. 48-58; Col. 7, ll. 10-24; 41-67; Col. 8, ll. 1-67; Col. 9, ll. 1-67; Col. 10, ll. 1-24, 66-67; and Col. 11, ll. 1-11.  U.S. Pat. No. 6,092,194 at	means-plus-function under § 112, ¶ 6  <b>Function:</b> receiving a Downloadable  <b>Corresponding structure:</b> indefinite for failure to disclose corresponding structure/algorithm  <u><b>Intrinsic Evidence:</b></u> '844 patent at Abstract, Figs. 4, 5, 7, 7:26-48, 9:20-23.  <u><b>Extrinsic Evidence:</b></u>  The following is a brief description of the testimony of Symantec's expert, Dr. Richard Ford, may offer regarding the phrase "means for receiving a Downloadable": 1) the technical background of the '844 patent; 2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>Figs. 1-5;                      Col. 1, ll. 60-67;                      Col. 2, ll. 1-36; 65-67;                      Col. 3, ll. 1-67;                      Col. 4, ll. 1-67;                      Col. 5, ll. 1-3; 15-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-6;                      Col. 9, ll. 57-67;                      Col. 10, ll. 1-6.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>March 2, 2015 Claim Construction Order – Finjan Inc. v. Sophos Inc., Civ. No. 3:14-cv-01197-WHO.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>3) how the phrase “means for receiving a Downloadable” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence, particularly, that the term would have been understood to be a means-plus-function limitation; and</p> <p>4) that the specification does not disclose a sufficient structure corresponding to the claimed function to one of ordinary skill in the art</p>
3.	means for generating a first Downloadable security profile that identifies	43	<p>Governed by 35 U.S.C. § 112(6):</p> <p><b>Function:</b> generating a first Downloadable security profile that identifies suspicious code in the received Downloadable</p>	<p>means-plus-function under § 112, ¶ 6</p> <p><b>Function:</b> generating a first Downloadable security profile that identifies suspicious code in the received Downloadable</p>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
	suspicious code in the received Downloadable		<p><b>Structure:</b> content inspection engine</p> <p><b><u>Intrinsic Evidence</u></b>                      Abstract;                      Figs. 1-8;                      Claims 10, 13, 15, 18, 19, 20, 21, 28, 36, 37, 43;                      Col. 2, ll. 3-60;                      Col. 4, ll. 59-64;                      Col. 5, ll. 48-58;                      Col. 7, ll. 10-24; 41-67;                      Col. 8, ll. 1-67;                      Col. 9, ll. 1-67;                      Col. 10, ll. 1-24, 66-67; and                      Col. 11, ll. 1-11.</p> <p>U.S. Pat. No. 6,092,194 at                      Figs. 1-5;                      Col. 1, ll. 60-67;                      Col. 2, ll. 1-36; 65-67;                      Col. 3, ll. 1-67;                      Col. 4, ll. 1-67;                      Col. 5, ll. 1-3; 15-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-6;                      Col. 9, ll. 57-67;                      Col. 10, ll. 1-6.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the</p>	<p><b>Corresponding structure:</b> a processor programmed to perform the algorithm disclosed at col. 5, lines 42-45 and col. 9, lines 20-42 of U.S. Patent No. 6,092,194</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’844 patent at Abstract, 3:66-4:19, 9:63-65, 4:35-36, 4:59-64, 8:47-64, Figs. 1, 4, 5, 7.</p> <p>U.S. Patent No. 6,092,194 at 5:41-57, 9:20-42, Figs. 6A, 7.</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable”:                      1) the technical background of the ’844 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence, particularly, that the term would have been understood to be a means-plus-function limitation; and                      4) the structure corresponding to the claimed</p>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.  March 2, 2015 Claim Construction Order – Finjan Inc. v. Sophos Inc., Civ. No. 3:14-cv-01197-WHO.  The intrinsic and extrinsic evidence cited by Defendant.	function disclosed by the specification to one of ordinary skill in the art.
4.	means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Structure:</b> content inspection engine  <u><b>Intrinsic Evidence</b></u> Abstract; Figs. 1-8; Claims 10, 13, 15, 18, 19, 20, 21, 28, 36, 37, 43; Col. 2, ll. 3-60; Col. 4, ll. 59-64; Col. 5, ll. 48-58; Col. 7, ll. 10-24; 41-67; Col. 8, ll. 1-67; Col. 9, ll. 1-67; Col. 10, ll. 1-24, 66-67; and Col. 11, ll. 1-11.  U.S. Pat. No. 6,092,194 at	means-plus-function under § 112, ¶ 6  <b>Function:</b> linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Corresponding structure:</b> a processor programmed to perform the algorithm of steps 630 and 645 disclosed at Fig. 6, col. 8 lines 65-67, col. 6, lines 13-24, and col. 5, lines 3-13.  <u><b>Intrinsic Evidence:</b></u> ’844 patent at Abstract, 2:5-7, 3:66-4:4, 4:35-36, 6:13-24, 8:49-67, Figs. 1, 4, 5, 6. 5/03/2000 Amendment and Response at 5-6.  <u><b>Extrinsic Evidence:</b></u> The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “means for linking

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>Figs. 1-5;                      Col. 1, ll. 60-67;                      Col. 2, ll. 1-36; 65-67;                      Col. 3, ll. 1-67;                      Col. 4, ll. 1-67;                      Col. 5, ll. 1-3; 15-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-6;                      Col. 9, ll. 57-67;                      Col. 10, ll. 1-6.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>March 2, 2015 Claim Construction Order – Finjan Inc. v. Sophos Inc., Civ. No. 3:14-cv-01197-WHO.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients”:</p> <p>1) the technical background of the ’844 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence, particularly, that the term would have been understood to be a means-plus-function limitation; and                      4) the structure corresponding to the claimed function disclosed by the specification to one of ordinary skill in the art.</p>
5.	before a web server makes the Downloadable available to web clients	1, 15, 41, 43	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      ‘844 patent at Abstract;                      Claims 1, 15, 22, 23, 32, 41, 42, 43, 44;</p>	<p>before [a/the] non-network gateway web server make[s] the Downloadable available to web clients</p> <p><b><u>Intrinsic Evidence:</u></b></p>



U.S. PATENT NO. 6,154,844				
Term	Claim(s)	Finjan’s Proposed Construction and Support		Defendant’s Proposed Construction and Support
		<p>Figs. 1-8;                      Col. 3, ll. 33-67;                      Col. 4, ll. 1-58;                      Col. 5, ll. 3-13, 59-65;                      Col. 8, ll.17-36;                      Col. 9, ll. 5-18;                      Col. 10, ll. 66-67;                      Col. 11, ll. 1-11.</p> <p>’844 Patent File History including: First Office Action on September 2, 1999 at 6-7; November 23, 1999 Response to Non-Final Office Action; February 8, 2000 Non-Final Rejection; Applicant Amendment on May16, 2000 at 5; and July 13, 2000 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>		<p>’844 patent at Figs. 1, 5, 6, 8; 5:3-13; 9:11-18, 10:24- 65.</p> <p>5/3/00 Amendment and Response at 5.</p> <p>U.S. Patent No. 7,418,731 at Figs. 1, 2, 1:25-60.</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>Order Construing Claims in U.S. Patent Nos. 6,154,844; 7,058,822; 7,418,731; 7,647,633, <i>Finjan Inc. v. Blue Coat Sys., Inc.</i>, No. 5:13-cv-3999-BLF (N.D. Cal.) (Dkt. No. 118).</p> <p>Finjan’s Claim Construction Opening Brief in <i>Finjan Inc. v. McAfee, Inc. et al.</i>, Case No. 10-cv-00593 (D. Del.) (Dkt. No. 144).</p> <p>Finjan’s Claim Construction Answering Brief in <i>Finjan Inc. v. McAfee, Inc. et al.</i>, Case No. 10-cv-00593 (D. Del.) (Dkt. No. 182).</p> <p>Finjan’s Opening Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 112).</p> <p>Finjan’s Answering Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 125).</p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “before a web server makes the</p>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
				Downloadable available to web clients”: 1) the technical background of the ’844 patent; 2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s); 3) how the phrase “before a web server makes the Downloadable available to web clients” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and 4) why Symantec’s proposed construction for this phrase is proper
6.	a first content inspection engine for using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable	15	No construction necessary – Plain and ordinary meaning. <u><b>Intrinsic Evidence</b></u> Abstract; Figs. 1-8; Claims 1-44; Col. 1, ll. 61-67; Col. 2, ll. 1-48; Col. 3, ll. 66-67; Col. 4, ll. 1-67; Col. 5, ll. 1-13; 48-58; Col. 7, ll. 10-25; 49-67; Col. 8, ll. 1-5; 17-36; 46-67; Col. 9, ll. 1-18; 54-67; and Col. 10, ll. 1-23.  ‘844 Patent File History including: September 2, 1999 Non-Final Rejection; November 23, 1999 Response to Non-Final Office Action;	means-plus-function under § 112, ¶ 6  <b>Function:</b> using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Corresponding structure:</b> a processor programmed to perform the algorithm disclosed at col. 5, lines 42-45 and col. 9, lines 20-42 of U.S. Patent No. 6,092,194 , and the algorithm of steps 630 and 645 disclosed at Fig. 6, col. 8 lines 65-67, col. 6, lines 13-24, and col. 5, lines 3-13 of the ‘844 patent.  <u><b>Intrinsic Evidence:</b></u> ‘844 patent at Abstract, 3:66-4:19, 9:63-65, 4:35-36,

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
	available to web clients		<p>February 8, 2000 Non-Final Rejection; May 16, 2000 Response to Non-Final Action; and July 13, 2000 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>4:59-64, 8:47-64, Figs. 1, 4, 5, 6, 7.</p> <p>U.S. Patent No. 6,092,194 at 5:41-57, 9:20-42, Figs. 6A, 7.</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “a first content inspection engine for using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’844 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the phrase “a first content inspection engine for using the first rule set to generate a first Downloadable security profile that identifies suspicious code in a Downloadable, and for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence, particularly, that the term would have been understood to be a means-plus-function limitation; and</li> <li>4) the structure corresponding to the claimed</li> </ol>

U.S. PATENT NO. 6,154,844				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
				function disclosed by the specification to one of ordinary skill in the art.

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
1.	database	1, 8, 15, 22, 29, 30	<p>a collection of interrelated data organized according to a database schema to serve one or more applications</p> <p><b><u>Intrinsic Evidence</u></b>                      US Patent No. 7,613,926 at Abstract; Figs. 2, 4, 7a, 7b; Claims 1, 8, 15, 22, 29, 30; Col. 3, ll. 49-62; Col. 8, ll. 17-64; Col. 9, ll. 49-62; Col. 11, ll. 41-57; Col. 12, ll. 3-13, 44-67; Col. 13, ll. 1-3, 25-38; Col. 14, ll. 56-63; and Col. 16, ll. 22-55.</p> <p>'926 Patent File History, including: February 25, 2009 Non-Final Rejection; May 26, 2009 Amendment and Response to Office Action; and August 6, 2009 Notice of Allowance.</p> <p>6,804,780 Patent at Abstract; Claim 18; Figs. 3, 4, 6A, 6B, 6C, 8; Col. 3, ll. 32-67; Col. 4, ll. 1-67; Col. 5, ll. 1-12; Col. 6, ll. 15-23; Col. 6, ll. 5-67; Col. 7, ll. 1-59; Col. 9, ll. 11-34, 58-67; and</p>	<p>organized collection of data</p> <p><b><u>Intrinsic Evidence:</u></b>                      '926 patent at 9:49-55.                      U.S. Patent No. 6,092,194 at 3:47-50, 4:14-18.                      Provisional Application No. 60/030,639 at 8-11.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Webster's Ninth New Collegiate Dictionary (9th Ed. 1992) at 325.                      The American Heritage Dictionary (3rd Ed. 1992) at 475.                      Random House Webster's College Dictionary (1999) at 339.                      21st Century Dictionary of Computer Terms (1994) at 95.                      Webster's New World Dictionary of Computer Terms (4th Ed. 1992) at 95.                      Microsoft Press Computer Dictionary (3d Ed. 1997) at 199-200, 403-404.                      Dictionary of Computer Words, Houghton Mifflin Company (1995) at 16, 108, 239-40.</p>

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>Col. 10, ll. 1-21.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p> <p>February 29, 2012 Order Construing the Terms of U.S. Patent Nos. 6,092,194 &amp; 6,480,962 – Finjan Inc. v. McAfee, Inc. et al. C.A. No. 10-cv-593 (GMS).</p> <p>August 12, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Websense, Inc., Civ. No. 13-cv-04398-BLF.</p> <p>March 2, 2015 Claim Construction Order – Finjan Inc. v. Sophos Inc., Civ. No. 3:14-cv-01197-WHO.</p> <p>IBM Dictionary of Computing, Tenth Edition, published 1994 by McGraw-Hill, Inc. and edited by George McDaniel.</p>	<p>U.S. Patent No. 6,513,047 at 2:30-31; 5:39-55; Fig. 5.</p> <p>U.S. Patent No. 5,857,190 at Abstract; Fig. 5; 3:7-46; 5:30-41; 8:1-64; 11:51-12:9; 12:10-23.</p> <p>C.J. Date, <i>An Introduction to Database Systems</i>, Addison-Wesley Publishing Company, at 2-9, 21-24, 52-53 (6th ed. 1995)</p> <p>Abraham Silberschatz et al., <i>Database System Concepts</i>, The McGraw-Hill Companies, Inc. (3d ed. 1997).</p> <p>Ramez Elmasri &amp; Shamkant B. Navathe, <i>Fundamentals of Database Systems</i>, at 23-37 (3d ed. 2000), available at <a href="https://ia700601.us.archive.org/11/items/FundamentalsOfDatabaseSystemselmasrinavathe/FundamentalsOfDatabaseSystemselmasrinavathe.pdf">https://ia700601.us.archive.org/11/items/FundamentalsOfDatabaseSystemselmasrinavathe/FundamentalsOfDatabaseSystemselmasrinavathe.pdf</a>.</p> <p><u>UNIX™ Time-Sharing System: UNIX Programmer’s Manual, Seventh Ed., Vol. 1, Bell Telephone Laboratories, Inc. (Jan. 1979).</u></p> <p><u>Barry Brachman &amp; Gerald Neufeld, TDBM: A DBM Library With Atomic Transactions, USENIX (June 8-12 1992).</u></p> <p>Gene H. Kim &amp; Eugene H. Spafford, <i>The Design and Implementation of Tripwire: A File System Integrity Checker</i>, Purdue University (Nov. 1993).</p> <p>Glenn Fowler, <i>cql – A Flat File Database Query</i></p>

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
			The intrinsic and extrinsic evidence cited by Defendant.	<p><i>Language</i>, AT&amp;T Research Laboratories (1994).</p> <p>Stephen Rauch, <i>Talk to Any Database the COM Way Using the OLE DB Interface</i>, Microsoft Systems Journal (July 1996).</p> <p>Dan S. Wallach et al., <i>Extensible Security Architectures for Java</i>, ACM (1997).</p> <p>Alok Sinha et al., <i>Behind the Scenes at MSN 2.0: Architecting an Internet-Based Online Service</i>, Microsoft Systems Journal (April 1997).</p> <p>getpwnam man page,  <a href="http://www.manpages.info/sunos/getpwnam.3.html">http://www.manpages.info/sunos/getpwnam.3.html</a>,                      last change May 18, 1999.</p> <p>The following is a brief description of the testimony of Symantec's expert, Dr. Richard Ford, may offer regarding the term "database":</p> <ol style="list-style-type: none"> <li>1) the technical background of the '926 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the term "database" would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</li> <li>4) why Symantec's proposed construction for this term is proper</li> </ol>
2.	Downloadable	1, 8, 15, 22, 29,	an executable application program, which is downloaded from a source computer and run on	mobile code that is requested by an ongoing process and downloaded from a source computer to a

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
		30	<p>the destination computer</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs. 1a, 1b, 1c, 3, 4, 5, 6b, 10a, 10b, 11, 12a, and 12b;                      Claims 1-13, 15-20, 22-27, 29-30;                      Col. 1, ll. 37-40, 51-67;                      Col. 2, ll. 1-20, 27-67;                      Col. 3, ll. 1-67;                      Col. 4, ll. 1-3, 19-35;                      Col. 5, ll. 38-62;                      Col. 6, ll. 3-44, 60-67;                      Col. 7, ll. 1-67;                      Col. 9, ll. 15-48, 63-67;                      Col. 10, ll. 1-9;                      Col. 11, ll. 4-23;                      Col. 12, ll. 14-23;                      Col. 14, ll. 56-67;                      Col. 15, ll. 1-14;                      Col. 16, ll. 56-67;                      Col. 17, ll. 1-18;                      Col. 19, ll. 23-67; and                      Col. 20, ll. 1-57.</p> <p>’926 Patent File History, including:                      February 25, 2009 Non-Final Rejection;                      May 26, 2009 Amendment and Response to Office Action; and                      August 6, 2009 Notice of Allowance.</p>	<p>destination computer for automatic execution</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’926 patent at Abstract, 1:8-32, 1:65-2:20, 6:3-18.                      U.S. Patent No. 6,154,844 at Abstract, 1:37-59, 1:62-2:2.                      U.S. Patent No. 6,092,194 file history, 10/27/1999 Preliminary Amendment at 6.                      U.S. Patent No. 6,092,194 file history, 1/3/2000 Notice of Allowance.                      Provisional Application No. 60/030,639 at 1-2, Appendix.                      U.S. Patent No. 6,804,780 file history, 7/31/2003 Amendment and Response to Office Action at 7.                      U.S. Patent No. 6,167,520 at 1:24-38.                      U.S. Patent No. 7,613,926 at 6:3-18.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Finjan’s Opening Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 112).                      Finjan’s Answering Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 125).                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “Downloadable”:</p>



<b>U.S. PATENT NO. 7,613,926</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>US 6,804,780 (“’780 Patent”) at:                      Title;                      Abstract;                      Claims 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18;                      Figs. 3, 4, 6A, 6B, 6C, 7, 8;                      Col. 1, ll. 50-67;                      Col. 2, ll. 1-16; 28-44;                      Col. 3, ll. 8-67;                      Col. 4, ll. 1-3, 50-67;                      Col. 5, ll. 1-24; 46-67;                      Col. 6, ll. 1-55;                      Col. 9, ll. 11-57; and                      Col. 10, ll. 6-21.</p> <p>’780 Patent File History including: Provisional Application No. 60/030,639; June 26, 2003 Non-Final Rejection; Office Action mailed July 1, 2003; Response to Non-Final Office Action August 4, 2003; Final Rejection mailed October 27, 2003; Request for Continued Examination February 27, 2004; Examiner’s Amendment Communication May 17, 2004.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the</p>	<p>1) the technical background of the ’926 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the term “Downloadable” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and                      4) why Symantec’s proposed construction for this term is proper</p>

<b>U.S. PATENT NO. 7,613,926</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p> <p>February 29, 2012 Order Construing the Terms of U.S. Patent Nos. 6,092,194 &amp; 6,480,962 – Finjan Inc. v. McAfee, Inc. et al. C.A. No. 10-cv-593 (GMS).</p> <p>August 12, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Websense, Inc., Civ. No. 13-cv-04398-BLF.</p> <p>October 14, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Sophos Inc., Civ. No. 14-cv-01197-WHO.</p> <p>October 20, 2014 Claim Construction Order – Finjan Inc. v. Blue Coat Systems, Inc., Civ. No. 5:13-cv-03999-BLF.</p> <p>January 26, 2015, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Proofpoint, Inc. et al., Civ. No. 13-cv-05808-BLF.</p>	

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			The intrinsic and extrinsic evidence cited by Defendant.	
3.	append[er/ed/ing]	1, 8, 29	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1a, 1b, 1c, 4, 6a, 6b, 10a, 10b, 11, 12a, and 12b;                      Claims 1-13, 15-20, 22-27, 29-30;                      Col. 1, ll. 37-40, 51-67;                      Col. 2, ll. 1-20, 27-67;                      Col. 3, ll. 1-67;                      Col. 4, ll. 1-3, 19-35;                      Col. 5, ll. 38-62;                      Col. 6, ll. 3-44, 60-67;                      Col. 7, ll. 1-67;                      Col. 9, ll. 15-48, 63-67;                      Col. 10, ll. 1-9;                      Col. 11, ll. 4-67;                      Col. 12, ll. 1-67;                      Col. 13, ll. 1-67;                      Col. 14, ll. 1-42; 56-67;                      Col. 15, ll. 1-14; 62-67;                      Col. 16, ll. 1-13; 56-67;                      Col. 17, ll. 1-18;                      Col. 19, ll. 23-67; and                      Col. 20, ll. 1-57.</p>	<p>attach to the end of</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’844 patent at Abstract, 2:5-7, 3:66-4:4, 4:35-36, 6:13-24, 8:49-67, Figs. 1, 4, 5, 6.                      5/03/2000 Amendment and Response at 5-6.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Microsoft Computer Dictionary (2nd Ed. 1994) at 22.                      21st Century Dictionary of Computer Terms (1994) at 12.                      Dictionary of Computer Words (Rev. Ed. 1995) at 8.                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “append[er/ed/ing]”:                      1) the technical background of the ’926 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</p>

U.S. PATENT NO. 7,613,926				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>‘926 Patent File History, including: February 25, 2009 Non-Final Rejection; May 26, 2009 Amendment and Response to Office Action; and August 6, 2009 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The American Heritage Dictionary (4th Ed. 2001) at 41.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>3) how the term “append[er/ed/ing]” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</p> <p>4) why Symantec’s proposed construction for this term is proper.</p>

U.S. PATENT NO. 7,756,996				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
1.	network gateway computer	1	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-4;                      Claims 1-3;                      Col. 1, ll. 38-67;                      Col. 2, ll. 1-61;                      Col. 3, ll. 4-67;                      Col. 4, ll. 1-67; and                      Col. 5, ll. 1-60.</p> <p>’996 Patent File History, including: January 2, 2009 Examiner Interview; January 7, 2010 Amendment and Response to Office Action; February 22, 2010 Examiner Interview; and March 3, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p>	<p>a computer that is a point of contact between different networks</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’996 patent at Fig. 2, 1:38-52, 3:4-16, 3:42-49, 4:30-40, 4:41-51,4:52-62, 4:63-67, 5:18-45.                      ’996 patent file history, 1/7/2010 Response and Amendment at 11.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Finjan’s Opening Claim Construction Brief in <i>Finjan Inc. v. Blue Coat Sys., Inc.</i>, No. 5:13-cv-3999-BLF (N.D. Cal.) (Dkt. No. 65).                      Finjan’s Opening Claim Construction Brief in <i>Finjan Inc. v. Sophos, Inc.</i>, No. 14-CV-01197-SBA (N.D. Cal.) (Dkt. No. 58).                      Dictionary of Computer and Internet Terms (11th Ed. 2013) at 217.                      Hutchinson Dictionary of Computing Multimedia and the Internet 3rd Ed. (3rd Ed. 1999) at 128.                      Newton’s Telecom Dictionary (19th Ed. 2003) at 350.                      Gardner’s Computer Graphics &amp; Animation Dictionary (2003) at 106.                      McGraw-Hill Dictionary of Scientific and Technical Terms (6th Ed. 2003) at 884</p>

U.S. PATENT NO. 7,756,996				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>October 20, 2014 Claim Construction Order – Finjan Inc. v. Blue Coat Systems, Inc., Civ. No. 5:13-cv-03999-BLF.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “network gateway computer”:</p> <p>1) the technical background of the ’996 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the term “network gateway computer” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and                      4) why Symantec’s proposed construction for this term is proper.</p>
2.	non-HTTP management data	1, 4, 7	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-4;                      Claims 1-3;                      Col. 1, ll. 38-67;                      Col. 2, ll. 1-61;                      Col. 3, ll. 4-67;                      Col. 4, ll. 1-67; and                      Col. 5, ll. 1-60.</p> <p>’996 Patent File History, including: January 2, 2009 Examiner Interview; January 7, 2010 Amendment and Response to Office Action; February 22, 2010 Examiner Interview; and</p>	<p>management data that the management server transmits and receives using a non-HTTP transport protocol</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’996 patent, Fig. 1, Fig. 2, Fig. 3, Fig. 4, 1:38-52, 1:29-32, 1:63-2:17,3:17-31, 3:50-57, 4:8-15, 4:52-67.                      ’996 patent file history, 1/7/2010 Response and Amendment at 2-12.                      ’996 patent file history, 2/22/2010 Examiner-Initiated Interview Summary.                      ’996 patent file history, 2/22/2010 Examiner’s Amendment and Reasons for Allowance at 2-7.</p>

<b>U.S. PATENT NO. 7,756,996</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>March 3, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “non-HTTP management data”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’996 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the term “non-HTTP management data” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</li> <li>4) why Symantec’s proposed construction for this term is proper.</li> </ol>

U.S. PATENT NO. 7,930,299				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
1.	dynamically updat[e/es/ing] the combined search and security results summary	1, 13, 20	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-10;                      Claims 1, 13, 20, 21;                      Col. 1, ll. 55-67;                      Col. 2, ll. 1-27; 47-67;                      Col. 3, ll. 1-29; and                      Col. 7, ll. 50-64.</p> <p>’299 Patent File History, including: December 8, 2008 Amendment and Response to Office Action; May 6, 2009 Amendment and Response to Office Action; February 1, 2010 Amendment and Response to Office Action; September 10, 2010 Amendment and Response to Office Action; and December 23, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p>	<p>updating the presented search results and assessments of potential security risks to present additional assessments of potential security risks after additional assessments of potential security risks are received</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’299 patent at 6:29-7:5, 7:6-37.                      9/10/2010 Response and Amendment at 11-15.                      12/23/2010 Notice of Allowance at 2-4.                      See also ’182 citations below.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Newton’s Telecom Dictionary (19th Ed. 2003) at 273.                      Microsoft Computer Dictionary (5th Ed. 2002) at 181.                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “dynamically updat[e/es/ing] the combined search and security results summary”:                      1) the technical background of the ’299 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the term “dynamically updat[e/es/ing] the</p>



<b>U.S. PATENT NO. 7,930,299</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			The intrinsic and extrinsic evidence cited by Defendant.	combined search and security results summary” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and  4) why Symantec’s proposed construction for this term is proper

U.S. PATENT NO. 8,015,182				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
1.	dynamically updating the presentation when additional security assessments are received	1, 8, 15	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-10;                      Claims 1, 8, 15                      Col. 2, ll. 11-67;                      Col. 3, ll. 1-13; 30-56;                      Col. 4, ll. 29-67;                      Col. 5, ll. 1-4;                      Col. 6, ll. 63-67;                      Col. 7, ll. 1-25; 35-52;                      Col. 8, ll. 30-40; 58-67;                      Col. 9, ll. 1-12; 25-62;                      Col. 10, ll. 1-9; 29-55;                      Col. 11, ll. 9-30; 44-67;                      Col. 12, ll. 1-3; 10-20; 39-67;                      Col. 13, ll. 1-67; and                      Col. 14, ll. 1-27.</p> <p>’182 Patent File History, including: December 10, 2008 Amendment and Response to Office Action; May 20, 2009 Amendment and Response to Office Action; December 1, 2009 Amendment and Response to Office Action; May 3, 2010 Amendment and Response to Office Action; August 2, 2010 Amendment and Response to Office Action; March 22, 2011 Amendment and Response to Office Action; and</p>	<p>updating the presentation of the search results summary and a portion of the security assessments to present additional security assessments when the additional security assessments are received</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’182 patent at 8:4-47, 8:48-9:12.                      3/22/2011 Response and Amendment at 9-11.                      6/27/2011 Notice of Allowance at 2-3.                      See also ’299 citations above.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Newton’s Telecom Dictionary (19th Ed. 2003) at 273.                      Microsoft Computer Dictionary (5th Ed. 2002) at 181.</p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “dynamically updating the presentation when additional security assessments are received”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’182 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the term “dynamically updating the presentation when additional security assessments are received” would have been understood by one</li> </ol>

<b>U.S. PATENT NO. 8,015,182</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>June 27, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</p> <p>4) why Symantec’s proposed construction for this term is proper.</p>

<b>U.S. PATENT NO. 7,757,289</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
1.	protecting a computer from dynamically generated malicious content	1, 10, 19, 22, 35, 41	<p>No construction necessary of preamble. If construed, plain and ordinary meaning should apply.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-5;                      Claims 1, 10, 19, 22, 25, 30, 35, 41                      Col. 5, ll. 13-61;                      Col. 6, ll. 13-67;                      Col. 7, ll. 1-7; 17-40; 52-67;                      Col. 8, ll. 20-67;                      Col. 9, ll. 1-3;                      Col. 13, ll. 66-67;                      Col. 14, ll. 1-8;                      Col. 15, ll. 37-53; and                      Col. 16, ll. 28-36.</p> <p>’289 Patent File History, including: July 9, 2009 Amendment; December 17, 2009 Amendment and Response to Office Action; and March 2, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct</p>	<p>Preamble is limiting.</p> <p>“protecting a computer from malicious content that is generated at run-time</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’289 patent at 3:18-24; 3:31-4:33; 4:37-63; 5:13-34; 5:35-61; 5:62-6:12; 6:13-23; 6:24-35; 6:44-58; 6:59-7:7; 7:17-28; 7:29-40; 7:53-59; 7:60-67; 8:48-54; 12:5-67; 13:32-35; 17:38-45.</p> <p>’154 patent file history, 10/5/2011 Office Action Response at 22-24.</p> <p>7/19/2011 Office Action Response in relation to Appl. No. 12/174192.</p> <p>U.S. Patent No. 5,983,348 (“Ji ‘348”) at Abstract.</p> <p>U.S. Patent No. 6,272,641 (“Ji ‘641”) at Abstract.</p> <p>U.S. Published App. 2007/0016948 (“Dubrovsky”) at ¶¶ 50-54.</p> <p><i>See also</i> ’154 citations below.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Dictionary of Computer Science, Engineering and Technology at 149 (2001).</p> <p>U.S. Patent No. 8,244,910 (“Davis”) (SYM-FIN0424153–SYM-FIN0424171) at 7:28-65.</p>

U.S. PATENT NO. 7,757,289				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			construction.  The intrinsic and extrinsic evidence cited by Defendant.	Halfond (SYM-FIN0424543–SYM-FIN0424549) at Abstract & §§ 1; 3.1; 3.2; 5.  The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “protecting a computer from dynamically generated malicious content”: 1) the technical background of the ’289 patent; 2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s); 3) how the phrase “protecting a computer from dynamically generated malicious content” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence and why one of ordinary skill in the art at the time of the invention would have understood the preamble to be limiting; and 4) why Symantec’s proposed construction for this phrase is proper
2.	content processor	10	No construction necessary – Plain and ordinary meaning.  <u><b>Intrinsic Evidence</b></u> Title; Abstract; Figs.1-5; Claims 10, 16; Col. 2, ll. 64-67; Col. 3, ll. 18-25; Col. 5, ll. 35-61; Col. 7, ll. 27-40; Col. 11, ll. 15-26;	software that renders the content for interactive viewing on a display monitor  <u><b>Intrinsic Evidence:</b></u> ’289 patent at Fig. 1; Fig. 2; Fig. 4; 2:64-3:2; 3:18-24; 3:41-4:3; 5:35-61; 7:28-40; 8:58-61; 10:45-11:26; 12:5-47; 13:25-31; 14:9-45; 15:63-67; 16:11-27.  ’154 patent file history, 10/5/11 Office Action Response at 18-19, 21-22, 22-24.

U.S. PATENT NO. 7,757,289				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>Col. 12, ll. 5-47;                      Col. 13, ll. 1-32;                      Col. 15, ll. 54-67; and                      Col. 16, ll. 11-27.</p> <p>‘289 Patent File History, including: July 9, 2009 Amendment; December 17, 2009 Amendment and Response to Office Action; and March 2, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p><i>See also</i> ‘154 citations below.</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “content processor”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ‘289 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the phrase “content processor” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</li> <li>4) why Symantec’s proposed construction for this phrase is proper</li> </ol>
3.	said content processor (i) suspends processing of the modified content after said client transmitter transmits the input to said security	16	<p>Not indefinite. Plain and ordinary meaning.</p> <p>To the extent a construction is required, the plain and ordinary meaning of this terms is:                      during processing of the modified content, said content processor first suspends processing of the modified content after said client transmitter transmits the input to said security computer, and then resumes processing of the</p>	<p>indefinite</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “said content processor (i) suspends processing of the modified content after said client transmitter transmits the input to said security computer, and (ii) resumes processing of</p>

U.S. PATENT NO. 7,757,289				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
	computer, and (ii) resumes processing of the modified content after said client receiver receives the indicator from said security computer		<p>modified content after said client receiver receives the indicator from said security computer.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-5;                      Claims 10, 16;                      Col. 2, ll. 64-67;                      Col. 3, ll. 18-25;                      Col. 5, ll. 35-61;                      Col. 7, ll. 27-40;                      Col. 11, ll. 15-26;                      Col. 12, ll. 5-47;                      Col. 13, ll. 1-32;                      Col. 15, ll. 54-67; and                      Col. 16, ll. 11-27.</p> <p>’289 Patent File History, including: July 9, 2009 Amendment; December 17, 2009 Amendment and Response to Office Action; and March 2, 2010 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct</p>	<p>the modified content after said client receiver”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’289 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s); and</li> <li>3) how the term “said content processor (i) suspends processing of the modified content after said client transmitter transmits the input to said security computer, and (ii) resumes processing of the modified content after said client receiver” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence.</li> </ol>

U.S. PATENT NO. 7,757,289				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			construction.  The intrinsic and extrinsic evidence cited by Defendant.	
4.	inspection[s]	1, 10, 19, 22, 35, 41	<p>No construction necessary – Plain and ordinary meaning</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-5;                      Claims 1, 10, 19, 22, 30, 35, 41                      Col. 4, ll. 4-63;                      Col. 5, ll. 13-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-52;                      Col. 8, ll. 20-44;                      Col. 9, ll. 47-67;                      Col. 10, ll. 1-67;                      Col. 11, ll. 1-31; 63-67;                      Col. 12, ll. 1-46;                      Col. 13, ll. 1-33; 43-52;                      Col. 14, ll. 25-45;                      Col. 15, ll. 6-23; 44-53;                      Col. 16, ll. 11-36; 52-62; and                      Col. 17, ll. 10-24.</p> <p>‘289 Patent File History, including: July 9, 2009 Amendment; December 17, 2009 Amendment and Response to Office Action; and March 2, 2010 Notice of Allowance.</p>	<p>scanning for the presence of potentially malicious operations</p> <p><b><u>Intrinsic Evidence:</u></b>                      ‘289 patent at Fig. 3, Fig. 5; 1:54-2:30; 3:18-4:33; 4:37-41; 11:32-13:31; 14:46-64; 15:6-22; 16:11-36; 17:10-24.                      ‘154 patent file history,                      10/5/2011 Office Action Response at 19-20, 22-24.  <i>See also</i> ‘154 citations below.</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “inspection[s]”:                      1) the technical background of the ‘289 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “inspection[s]” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</p>



<b>U.S. PATENT NO. 7,757,289</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>4) why Symantec’s proposed construction for this phrase is proper.</p>

<b>U.S. PATENT NO. 8,677,494</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
1.	Downloadable	1, 10	<p>an executable application program, which is downloaded from a source computer and run on the destination computer</p> <p>Title;                      Abstract;                      Figs. 1a, 1b, 1c, 3, 4, 5, 7b, 9, 10A, 11, 12a, 12b;                      Claims 1-18;                      Col. 2, ll. 8-2:47, 41-67;                      Col. 5, ll. 1-5, 60-67;                      Col. 6, ll. 1-6, 26-55;                      Col. 7, ll. 16-67;                      Col. 8, ll. 1-30;                      Col. 9, ll. 38-52;                      Col. 11, ll. 8-16, 28-47;                      Col. 12, ll. 19-26, 38-67;                      Col. 13, ll. 1-8, 49-67;                      Col. 14, ll. 1-28;                      Col. 15, ll. 14-64;                      Col. 16, ll. 5-20, 48-67;                      Col. 17, ll. 59-67;                      Col. 18, ll. 1-6, 56-67;                      Col. 19, ll. 1-2, 16-67;                      Col. 20, ll. 1-12, 35-67; and                      Col. 21, ll. 1-17.</p> <p>’494 Patent File History including: Application No. 13/290,708; November 7, 2011 Applicant’s Preliminary Amendment and Remarks; July 23, 2013 Office Action – Non-Final Rejection; October 23, 2012 Amendment and Response to Office Action; January 7, 2013 Office Action – Final Rejection; May 7, 2013 Amendment and Response to Office</p>	<p>mobile code that is requested by an ongoing process and downloaded from a source computer to a destination computer for automatic execution</p> <p><b><u>Intrinsic Evidence:</u></b></p> <p>’494 patent at Abstract, 1:8-55, 2:22-44,2:51-3:2, 6:3-18.</p> <p>U.S. Patent No. 6,154,844 at Abstract, 1:37-59, 1:62-2:2.</p> <p>U.S. Patent No. 6,092,194 file history, 10/27/1999 Preliminary Amendment at 6.</p> <p>U.S. Patent No. 6,092,194 file history, 1/3/2000 Notice of Allowance.</p> <p>Provisional Application No. 60/030,639 at 1-2, Appendix.</p> <p>U.S. Patent No. 6,804,780 prosecution history, 7/31/2003 Amendment and Response to Office Action at 7.</p> <p>U.S. Patent No. 6,167,520 at 1:24-38.</p> <p>U.S. Patent No. 7,613,926 at 6:3-18.</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>Finjan’s Opening Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 112).</p>

<b>U.S. PATENT NO. 8,677,494</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>Action under 37 C.F.R. 1.114; May 7, 2013 Declaration of Prior Invention in the United States to Overcome Cited Patent or Publication; and August 29, 2013 Notice of Allowance.</p> <p>All citations to intrinsic evidence for the term “Downloadable” for the ‘780 Patent, which the ‘494 Patent incorporates by reference.</p> <p><u>Extrinsic Evidence</u>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p> <p>February 29, 2012 Order Construing the Terms of U.S. Patent Nos. 6,092,194 &amp; 6,480,962 – Finjan Inc. v. McAfee, Inc. et al. C.A. No. 10-cv-593 (GMS).</p> <p>August 12, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Websense, Inc., Civ. No. 13-cv-04398-BLF.</p>	<p>Finjan’s Answering Claim Construction Brief in <i>Finjan Software, Ltd. V. Secure Computing Corp.</i>, Case No. 06-cv-00369 (D. Del.) (Dkt. No. 125).</p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “Downloadable”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ‘926 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</li> <li>3) how the term “Downloadable” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</li> <li>4) why Symantec’s proposed construction for this term is proper.</li> </ol>

U.S. PATENT NO. 8,677,494				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>October 14, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Sophos Inc., Civ. No. 14-cv-01197-WHO.</p> <p>October 20, 2014 Claim Construction Order – Finjan Inc. v. Blue Coat Systems, Inc., Civ. No. 5:13-cv-03999-BLF.</p> <p>January 26, 2015, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Proofpoint, Inc. et al., Civ. No. 13-cv-05808-BLF.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	
2.	database	1, 10	<p>a collection of interrelated data organized according to a database schema to serve one or more applications</p> <p><b><u>Intrinsic Evidence</u></b></p> <p>Abstract;                      Claims 1, 2, 10, 11;                      Figs. 2, 4, 7a, 7b;                      Col. 3, ll. 16-52;                      Col. 8, ll. 31-67;                      Col. 9, ll. 1-7;                      Col. 10, ll. 5-19;                      Col. 11, ll. 65-67;                      Col. 12, ll. 1-14;                      Col. 17, ll. 1-29; and</p>	<p>organized collection of data</p> <p><b><u>Intrinsic Evidence:</u></b></p> <p>U.S. Patent No. 6,092,194 at 3:47-50, 4:14-18.                      Provisional Application No. 60/030,639 at 8-11.</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>Webster’s Ninth New Collegiate Dictionary (9th Ed. 1992) at 325.                      The American Heritage Dictionary (3rd Ed. 1992) at 475.                      Random House Webster’s College Dictionary (1999) at 339.</p>

<b>U.S. PATENT NO. 8,677,494</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>Col. 18, ll. 7-15.</p> <p>‘494 Patent File History including: Application No. 13/290,708; November 7, 2011 Applicant’s Preliminary Amendment and Remarks; July 23, 2013 Office Action – Non-Final Rejection; October 23, 2012 Amendment and Response to Office Action; January 7, 2013 Office Action – Final Rejection; May 7, 2013 Amendment and Response to Office Action under 37 C.F.R. 1.114; May 7, 2013 Declaration of Prior Invention in the United States to Overcome Cited Patent or Publication; and August 29, 2013 Notice of Allowance.</p> <p>’780 Patent at                      Abstract;                      Claim 18;                      Figs. 3, 4, 6A, 6B, 6C, 8;                      Col. 3, ll. 32-67;                      Col. 4, ll. 1-67;                      Col. 5, ll. 1-12;                      Col. 6, ll. 15-23;                      Col. 6, ll. 5-67;                      Col. 7, ll. 1-59;                      Col. 9, ll. 11-34, 58-67; and                      Col. 10, ll. 1-21.</p> <p><u>Extrinsic Evidence</u>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the</p>	<p>21st Century Dictionary of Computer Terms (1994) at 95.</p> <p>Webster’s New World Dictionary of Computer Terms (4th Ed. 1992) at 95.</p> <p>Microsoft Press Computer Dictionary (3d Ed. 1997) at 199-200, 403-404.</p> <p>Dictionary of Computer Words, Houghton Mifflin Company (1995) at 16, 108, 239-40.</p> <p>U.S. Patent No. 6,513,047 at 2:30-31; 5:39-55; Fig. 5.</p> <p>U.S. Patent No. 5,857,190 at Abstract; Fig. 5; 3:7-46; 5:30-41; 8:1-64; 11:51-12:9; 12:10-23.</p> <p>C.J. Date, <i>An Introduction to Database Systems</i>, Addison-Wesley Publishing Company, at 2-9, 21-24, 52-53 (6th ed. 1995)</p> <p>Abraham Silberschatz et al., <i>Database System Concepts</i>, The McGraw-Hill Companies, Inc. (3d ed. 1997).</p> <p>Ramez Elmasri &amp; Shamkant B. Navathe, <i>Fundamentals of Database Systems</i>, at 23-37 (3d ed. 2000), available at <a href="https://ia700601.us.archive.org/11/items/FundamentalsOfDatabaseSystemselmasrinavathe/FundamentalsOfDatabaseSystemselmasrinavathe.pdf">https://ia700601.us.archive.org/11/items/FundamentalsOfDatabaseSystemselmasrinavathe/FundamentalsOfDatabaseSystemselmasrinavathe.pdf</a>.</p> <p><u>UNIX™ Time-Sharing System: UNIX Programmer’s Manual, Seventh Ed., Vol. 1, Bell</u></p>

<b>U.S. PATENT NO. 8,677,494</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
			<p>perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>December 11, 2007 Claim Construction Order – Finjan Software, Ltd. v. Secure Computer Corp. et al., C.A. No. 06-269.</p> <p>February 29, 2012 Order Construing the Terms of U.S. Patent Nos. 6,092,194 &amp; 6,480,962 – Finjan Inc. v. McAfee, Inc. et al. C.A. No. 10-cv-593 (GMS).</p> <p>August 12, 2014, Joint Claim Construction and Pre-Hearing Statement Pursuant to Patent Local Rule 4-3, Finjan, Inc. v. Websense, Inc., Civ. No. 13-cv-04398-BLF.</p> <p>March 2, 2015 Claim Construction Order – Finjan Inc. v. Sophos Inc., Civ. No. 3:14-cv-01197-WHO.</p> <p>IBM Dictionary of Computing, Tenth Edition, published 1994 by McGraw-Hill, Inc. and edited by George McDaniel.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p><u>Telephone Laboratories, Inc. (Jan. 1979).</u></p> <p><u>Barry Brachman &amp; Gerald Neufeld, <i>TDBM: A DBM Library With Atomic Transactions</i>, USENIX (June 8-12 1992).</u></p> <p>Gene H. Kim &amp; Eugene H. Spafford, <i>The Design and Implementation of Tripwire: A File System Integrity Checker</i>, Purdue University (Nov. 1993).</p> <p>Glenn Fowler, <i>cql – A Flat File Database Query Language</i>, AT&amp;T Research Laboratories (1994).</p> <p>Stephen Rauch, <i>Talk to Any Database the COM Way Using the OLE DB Interface</i>, Microsoft Systems Journal (July 1996).</p> <p>Dan S. Wallach et al., <i>Extensible Security Architectures for Java</i>, ACM (1997).</p> <p>Alok Sinha et al., <i>Behind the Scenes at MSN 2.0: Architecting an Internet-Based Online Service</i>, Microsoft Systems Journal (April 1997).</p> <p>getpwnam man page, <a href="http://www.manpages.info/sunos/getpwnam.3.html">http://www.manpages.info/sunos/getpwnam.3.html</a>, last change May 18, 1999.</p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the term “database”:</p> <p>1) the technical background of the ’494 patent;</p>

<b>U.S. PATENT NO. 8,677,494</b>				
	<b>Term</b>	<b>Claim(s)</b>	<b>Finjan’s Proposed Construction and Support</b>	<b>Defendant’s Proposed Construction and Support</b>
				2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);  3) how the term “database” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and  4) why Symantec’s proposed construction for this term is proper.

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
1.	protecting a computer from dynamically generated malicious content	1, 6	<p>No construction necessary of preamble. If construed, plain and ordinary meaning should apply.</p> <p><b><u>Intrinsic Evidence</u></b>                      Title;                      Abstract;                      Figs.1-5;                      Claims 1, 6                      Col. 5, ll. 4-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-67;                      Col. 8, ll. 1-60;                      Col. 13, ll. 37-46;                      Col. 15, ll. 8-23; 65-67; and                      Col. 16, ll. 1-6.</p> <p>'154 Patent File History, including:                      June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office Action; and December 22, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the</p>	<p>Preamble is limiting.</p> <p>protecting a computer from malicious content that is generated at run-time</p> <p><b><u>Intrinsic Evidence:</u></b>                      '154 Patent at 3:18-24; 3:31-4:26; 4:30-54; 5:4-25; 5:26-52; 5:53-6:3; 6:4-14; 6:15-26; 6:35-49; 6:50-65; 7:9-19; 7:20-31; 7:44-50; 7:51-58; 8:38-44; 11:50-12:42; 13:4-7; 17:8-15.</p> <p>10/5/2011 Office Action Response at 22-24.</p> <p>7/19/2011 Office Action Response in relation to Appl. No. 12/174192.</p> <p>U.S. Patent No. 5,983,348 ("Ji '348") at Abstract.</p> <p>U.S. Patent No. 6,272,641 ("Ji '641") at Abstract.</p> <p>U.S. Published App. 2007/0016948 ("Dubrovsky") at ¶¶ 50-54.</p> <p><i>See also</i> '289 citations above.</p> <p><b><u>Extrinsic Evidence:</u></b>                      Dictionary of Computer Science, Engineering and Technology at 149 (2001).</p> <p>U.S. Patent No. 8,244,910 ("Davis") (SYM-FIN0424153-SYM-FIN0424171) at 7:28-65.</p>



U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
			<p>intrinsic record and extrinsic evidence, and why Finjan's construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>Halfond (SYM-FIN0424543–SYM-FIN0424549) at Abstract &amp; §§ 1; 3.1; 3.2; 5.</p> <p>The following is a brief description of the testimony of Symantec's expert, Dr. Richard Ford, may offer regarding the phrase "protecting a computer from dynamically generated malicious content":</p> <p>1) the technical background of the '154 patent;</p> <p>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);</p> <p>3) how the phrase "protecting a computer from dynamically generated malicious content" would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence and why one of ordinary skill in the art at the time of the invention would have understood the preamble to be limiting; and</p> <p>4) why Symantec's proposed construction for this phrase is proper</p>
2.	content processor	1, 6	<p>No construction necessary – Plain and ordinary meaning</p> <p><b><u>Intrinsic Evidence</u></b>                      Abstract;                      Figs. 1-5;                      Claims 1-3, 6-8;                      Col. 2, ll. 54-67;                      Col. 3, ll. 1-24;                      Col. 5, ll. 25-52;                      Col. 6, ll. 50-65;</p>	<p>software that renders the content for interactive viewing on a display monitor</p> <p><b><u>Intrinsic Evidence:</u></b>                      '154 patent at Fig. 1; Fig. 2; Fig. 4; 2:64-3:2; 3:18-24; 3:40-64; 5:26-52; 7:20-31; 8:49-51; 10:30-11:4; 11:50-12:24; 12:64-13:3; 13:47-14:16; 15:33-37; 15:48-64.                      10/5/2011 Office Action Response at 18-19, 21-22,</p>

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>Col. 9, ll. 5-12;                      Col. 10, ll. 60-67;                      Col. 11, ll. 1-67;                      Col. 12, ll. 1-67;                      Col. 13, ll. 1-4; and                      Col. 15, ll. 33-64.</p> <p>‘154 Patent File History, including:                      June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office Action; and December 22, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>22-24.</p> <p><i>See also</i> ‘289 citations above.</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “content processor”:                      1) the technical background of the ‘154 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “content processor” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and                      4) why Symantec’s proposed construction for this phrase is proper.</p>
3.	said content processor (i) suspends processing of the content after said transmitter transmits the	2, 7	<p>Not indefinite. Plain and ordinary meaning.</p> <p>To the extent a construction is required, the plain and ordinary meaning of this terms is:                      during processing content received over a network, said content processor first suspends processing of the content after said transmitter</p>	<p>indefinite</p> <p><b><u>Extrinsic Evidence:</u></b></p> <p>The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford,</p>

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
	input to the security computer, and (ii) resumes processing of the content after said receiver receives the [indicator/modified input variable]		<p>transmits the input to the security computer, and then resumes processing of the content after said receiver receives the [indicator/modified input variable].</p> <p><b><u>Intrinsic Evidence</u></b>                      Abstract;                      Figs. 1-5;                      Claims 1-12;                      Col. 2, ll. 54-67;                      Col. 3, ll. 1-24;                      Col. 5, ll. 25-52;                      Col. 6, ll. 50-65;                      Col. 9, ll. 5-12;                      Col. 10, ll. 60-67;                      Col. 11, ll. 1-67;                      Col. 12, ll. 1-67;                      Col. 13, ll. 1-4; and                      Col. 15, ll. 33-64.</p> <p>‘154 Patent File History, including: June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office Action; and December 22, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the</p>	<p>may offer regarding the term “said content processor (i) suspends processing of the content after said transmitter transmits the input to the security computer, and (ii) resumes processing of the content after said receiver receives the [indicator/modified input variable]”:</p> <ol style="list-style-type: none"> <li>1) the technical background of the ’154 patent;</li> <li>2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s); and</li> <li>3) how the term “said content processor (i) suspends processing of the content after said transmitter transmits the input to the security computer, and (ii) resumes processing of the content after said receiver receives the [indicator/modified input variable]” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence.</li> </ol>

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
			intrinsic record and extrinsic evidence, and why Finjan's construction is the correct construction.  The intrinsic and extrinsic evidence cited by Defendant.	
4.	[invoking / invoke / calling] a second function	1, 4, 6, 10	No construction necessary – Plain and ordinary meaning.  <u><b>Intrinsic Evidence</b></u> Abstract; Figs. 1-5; Claims 1-12; Col. 7, ll. 20-43; Col. 9, ll. 36-67; Col. 10, ll. 1-60; Col. 11, ll. 40-67; Col. 12, ll. 1-67; and Col. 13, ll. 1-4.  '154 Patent File History, including: June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office Action; and December 22, 2011 Notice of Allowance.  <u><b>Extrinsic Evidence</b></u> Testimony from Dr. Nenad Medvidovic regarding the proper construction of the term from the perspective of one of skill in the art	[invoking / invoke / calling] a function different from the first function  <u><b>Intrinsic Evidence:</b></u> '154 patent at Abstract; 5:4-52, Fig. 2 and corresponding text; Fig. 5 and corresponding text; 6:4-26; 5:53-6:3; 7:8-7:43; 10:1-14; 12:42 – 13:2; 14:17-29; 15:7-14; 4:55-4:60; 9:5-13; 9:13-30; 9:5-11:10; Claim 6; Claim 10.  10/5/2011 Amendment and Response to Office Action at 14-15, 18-19, 22-24.  <u><b>Extrinsic Evidence:</b></u> The following is a brief description of the testimony of Symantec's expert, Dr. Richard Ford, may offer regarding the phrase "[invoking/invoke/calling] a second function": 1) the technical background of the '154 patent; 2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s); 3) how the phrase "[invoking/invoke/ calling] a second function" would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>based on the intrinsic record and extrinsic evidence.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>evidence and why one of ordinary skill in the art at the time of the invention would have understood the preamble to be limiting; and</p> <p>4) why Symantec’s proposed construction for this phrase is proper..</p>
5.	a call to a first function	1, 4, 6, 10	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Abstract;                      Figs. 1-5;                      Claims 1-12;                      Col. 7, ll. 20-43;                      Col. 9, ll. 36-67;                      Col. 10, ll. 1-60;                      Col. 11, ll. 40-67;                      Col. 12, ll. 1-67; and                      Col. 13, ll. 1-4.</p> <p>’154 Patent File History, including:                      June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office Action; and December 22, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the</p>	<p>a call to a function different from the second function</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’154 patent at Abstract; 5:4-52, Fig. 2 and corresponding text; Fig. 5 and corresponding text; 6:4-26; 5:53-6:3; 7:8-7:43; 10:1-14; 12:42 – 13:2; 14:17-29; 15:7-14; 4:55-4:60; 9:5-13; 9:13-30; 9:5-11:10; Claim 6; Claim 10.</p> <p>10/5/2011 Amendment and Response to Office Action at 14-15, 18-19, 22-24.</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “a call to a first function”:                      1) the technical background of the ’154 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “a call to a first function” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence and why one of</p>

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan’s Proposed Construction and Support	Defendant’s Proposed Construction and Support
			<p>perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan’s construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>ordinary skill in the art at the time of the invention would have understood the preamble to be limiting; and</p> <p>4) why Symantec’s proposed construction for this phrase is proper.</p>
6.	inspection	1, 4, 6, 10	<p>No construction necessary – Plain and ordinary meaning.</p> <p><b><u>Intrinsic Evidence</u></b>                      Abstract;                      Figs. 1-5;                      Claims 1-12;                      Col. 3, ll. 65-67;                      Col. 4, ll. 1-67;                      Col. 5, ll. 1-67;                      Col. 6, ll. 1-67;                      Col. 7, ll. 1-43;                      Col. 8, ll. 10-34;                      Col. 9, ll. 36-67;                      Col. 10, ll. 1-67;                      Col. 11, ll. 1-10; 40-67;                      Col. 12, ll. 7-24; 43-67;                      Col. 13, ll. 1-23;                      Col. 14, ll. 1-16; 44-60;                      Col. 15, ll. 14-23; 48-67; and                      Col. 16, ll. 1-6; 22-32; 47-61.</p> <p>’154 Patent File History, including:                      June 28, 2011 Non-Final Rejection; October 5, 2011 Amendment and Response to Office</p>	<p>Scanning for the presence of potentially malicious operations</p> <p><b><u>Intrinsic Evidence:</u></b>                      ’154 patent at Fig. 3; Fig. 5; 1:54-2:30; 3:18-4:26; 4:30-34; 11:10-13:3; 14:17-14:35; 14:44-60; 15:48-16:6; 16:47-61.</p> <p>10/5/2011 Office Action Response at 19-20, 22-24.</p> <p><i>See also</i> ’289 citations above.</p> <p><b><u>Extrinsic Evidence:</u></b>                      The following is a brief description of the testimony of Symantec’s expert, Dr. Richard Ford, may offer regarding the phrase “inspection”:                      1) the technical background of the ’154 patent;                      2) the qualifications of one of ordinary skill in the art at the time of the alleged invention(s);                      3) how the phrase “inspection” would have been understood by one of ordinary skill in the art at the time of the invention(s) in light of the intrinsic and extrinsic evidence; and</p>

U.S. PATENT NO. 8,141,154				
	Term	Claim(s)	Finjan's Proposed Construction and Support	Defendant's Proposed Construction and Support
			<p>Action; and December 22, 2011 Notice of Allowance.</p> <p><b><u>Extrinsic Evidence</u></b>                      Testimony from Dr. Nenad Medvidovic regarding the technical background of the patent, the qualifications of one of skill in the art, the proper function and structure from the perspective of one of skill in the art based on the intrinsic record and extrinsic evidence, and why Finjan's construction is the correct construction.</p> <p>The intrinsic and extrinsic evidence cited by Defendant.</p>	<p>4) why Symantec's proposed construction for this phrase is proper</p>

# **EXHIBIT 10**



1 PAUL J. ANDRE (State Bar No. 196585)  
pandre@kramerlevin.com  
2 LISA KOBIALKA (State Bar No. 191404)  
lkobialka@kramerlevin.com  
3 JAMES HANNAH (State Bar No. 237978)  
jhannah@kramerlevin.com  
4 KRAMER LEVIN NAFTALIS & FRANKEL  
5 LLP  
990 Marsh Road  
6 Menlo Park, CA 94025  
Telephone: (650) 752-1700  
7 Facsimile: (650) 752-1800  
8 *Attorneys for Plaintiff*  
FINJAN, INC.

SEAN C. CUNNINGHAM (Bar No. 174931)  
sean.cunningham@dlapiper.com  
KATHRYN RILEY GRASSO (Bar No. 211187)  
kathryn.riley@dlapiper.com  
DLA PIPER LLP (US)  
401 B Street, Suite 1700  
San Diego, CA 92101-4297  
Telephone: (619) 699-2700  
Facsimile: (619) 699-2701  
  
Attorneys for Defendant and Counterclaim  
Plaintiff SOPHOS, INC and Counterclaim  
Plaintiff SOPHOS LTD.

10  
11 **IN THE UNITED STATES DISTRICT COURT**  
12 **FOR THE NORTHERN DISTRICT OF CALIFORNIA**  
13 **SAN FRANCISCO DIVISION**

14 FINJAN, INC., a Delaware Corporation,  
15 Plaintiff,  
16 v.  
17 SOPHOS INC., a Massachusetts Corporation,  
18 Defendant.

Case No.: 14-CV-01197-WHO

**JOINT CLAIM CONSTRUCTION AND  
PRE-HEARING STATEMENT  
PURSUANT TO PATENT LOCAL RULE  
4-3**

Pursuant to the Court’s Case Management Order and Patent L.R. 4-3 Plaintiff Finjan, Inc. (“Finjan”) and Defendant Sophos, Inc. (“Sophos”) hereby submit this Joint Claim Construction and Pre-Hearing Statement.

**I. PATENT L.R. 4-3(a): CLAIM TERMS ON WHICH THE PARTIES AGREE.**

During the meet and confer process, the parties have agreed to the following constructions:

Claim Term	Agreed Construction
Downloadable	an executable application program, which is downloaded from a source computer and run on the destination computer
CODE-A	potentially malicious executable code
CODE-B	executable wrapper code

**II. PATENT L.R. 4-3(b): PROPOSED CONSTRUCTION OF EACH DISPUTED TERM.**

The parties’ proposed claim constructions are provided below. All supporting evidence for the parties’ claim constructions is provided in Exhibit A. The parties reserve their rights to cite additional supporting evidence based on arguments raised in the claim construction briefs.

U.S. Patent No. 6,154,844			
Claim Term	Claim(s)	Finjan’s Proposed Construction	Sophos’s Proposed Construction
means for receiving a Downloadable	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> receiving a Downloadable  <b>Structure:</b> Downloadable file interceptor	Indefinite
means for generating a first Downloadable security profile that identifies suspicious code in the received	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> generating a first Downloadable security profile that identifies suspicious code	Indefinite

U.S. Patent No. 6,154,844			
Claim Term	Claim(s)	Finjan's Proposed Construction	Sophos's Proposed Construction
Downloadable		in the received Downloadable  <b>Structure:</b> content inspection engine	
means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients	43	Governed by 35 U.S.C. § 112(6):  <b>Function:</b> linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients  <b>Structure:</b> content inspection engine	Indefinite

U.S. Patent No. 7,613,918			
Claim Term	Claim(s)	Finjan's Proposed Construction	Sophos's Proposed Construction
CODE-C	12, 22	combined code	combined code created at the gateway computer
security context	12, 22	No construction necessary	an environment in which a software application is run, which may limit resources that the application is permitted to access or operations that the application is permitted to perform

U.S. Patent No. 7,613,926			
Claim Term	Claim(s)	Finjan's Proposed Construction	Sophos's Proposed Construction
database	22	a collection of interrelated data organized according to a database schema to serve one or more applications	no construction necessary

U.S. Patent No. 8,566,580			
Claim Term	Claim(s)	Finjan's Proposed Construction	Sophos's Proposed Construction
certificate creator	1	No construction necessary	security gateway component that creates a signed certificate for attributes of a server certificate
protocol appender	1	No construction necessary	an apparatus that appends certificate attributes within a protocol request

U.S. Patent No. 8,677,494			
Claim Term	Claim(s)	Finjan's Proposed Construction	Sophos's Proposed Construction
database	1, 10	a collection of interrelated data organized according to a database schema to serve one or more applications	No construction necessary

**III. PATENT L.R. 4-3(c): IDENTIFICATION OF MOST SIGNIFICANT TERMS.**

FINJAN'S STATEMENT:

Finjan does not consider any of the disputed terms significant or case or claim dispositive.

SOPHOS'S STATEMENT:

Sophos believes the terms the following terms are significant because their indefiniteness invalidates their respective claims:

- “means for receiving a Downloadable” - '844 patent (claim 43)
- “means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable” - '844 patent (claim 43)
- “means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients” - '844 patent (claim 43).

**IDENTIFICATION OF 8 CLAIM TERMS FOR CLAIM CONSTRUCTION BRIEFING:**

The parties select the following 8 terms for briefing:

No.	Patent(s)	Term
1.	'844: 43	means for receiving a Downloadable
2.	'844: 43	means for generating a first Downloadable security profile that identifies suspicious code in the received Downloadable
3.	'844: 43	means for linking the first Downloadable security profile to the Downloadable before a web server makes the Downloadable available to web clients
4.	'918: 12, 22	security context
5.	'918: 12, 22	CODE-C
6.	'926: 22 '494: 1, 10	database
7.	'580: 1	certificate creator
8.	'580: 1	protocol appender

**IV. PATENT L.R. 4-3(d): TIME FOR CLAIM CONSTRUCTION HEARING.**

The parties anticipate that they will not require more than 4 hours for the entire claim construction hearing.

**V. PATENT L.R. 4-3(e): WITNESSES AT CLAIM CONSTRUCTION HEARING.****Finjan's Statement:**

Finjan intends to offer a declaration and may present live witness testimony from Dr. Nenad Medvidovic, University of Southern California, 941 Bloom Walk, Los Angeles, CA 90089, to support Finjan's claim construction positions. Furthermore, Dr. Nenad Medvidovic will offer an opinion regarding the definiteness of the claims. Finjan may also present live testimony from Dr. Nenad Medvidovic in conjunction with a tutorial of the technology.



1  
2 Respectfully submitted,

3 Dated: October 14, 2014

4 By: /s/ Sean C. Cunningham

5 SEAN C. CUNNINGHAM  
6 sean.cunningham@dlapiper.com  
7 KATHRYN RILEY GRASSO  
8 kathryn.riley@dlapiper.com  
9 DLA PIPER LLP (US)  
10 401 B Street, Suite 1700  
11 San Diego, CA 92101-4297  
12 Telephone: (619) 699-2700  
13 Facsimile: (619) 699-2701

14 RYAN W. COBB (Bar No. 277608)  
15 ryan.cobb@dlapiper.com  
16 SUMMER KRAUSE (Bar No. 264858)  
17 summer.krause@dlapiper.com  
18 DLA PIPER LLP (US)  
19 2000 University Avenue  
20 East Palo Alto, CA 94303-2215  
21 Tel: (650) 833-2000  
22 Fax: (650) 833-2001

23 *Attorneys for Defendant and Counterclaim*  
24 *Plaintiff SOPHOS, INC and Counterclaim*  
25 *Plaintiff SOPHOS LTD.*

26  
27 In accordance with Civil Local Rule 5-1(i)(3), I attest that concurrence in the filing of this  
28 document has been obtained from any other signatory to this document.

By: /s/ James Hannah  
James Hannah

**EXHIBIT 11**

---



*Inter Partes* Review of  
U.S. Patent No. 7,613,926

Filed on behalf of Symantec Corporation

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL  
AND APPEAL BOARD

---

SYMANTEC CORPORATION

Petitioner

v.

FINJAN, INC.

Patent Owner

---

Case To Be Assigned  
U.S. Patent No. 7,613,926

---

**DECLARATION OF JACK W. DAVIDSON IN SUPPORT OF  
PETITIONER PURSUANT TO 37 C.F.R. § 42.120**

Symantec 1015  
IPR of U.S. Pat. No. 7,613,926

simply transmitting the DSP itself (e.g., attaching a copy of the DSP to the Downloadable).

88. In my opinion, it is clear that the ‘926 patent does not disclose anything new with respect to generating security profile data (i.e., a list of potentially suspicious operations) for an executable. Instead the patent relies on well-known techniques for deriving such data, such as by parsing and decomposing executable code. ‘194 patent, col. 5:42-45. Thus, the only alleged differences in the challenged claims appears to be that a hash of the Downloadable is used to retrieve the DSP from a database and a representation of this DSP (as well as the Downloadable itself) is sent to the intended recipient. ‘926 patent, col. 21:58-22:4, claim 15. In my opinion one of ordinary skill in the art would have found these concepts, i.e., using a hash to retrieve information from a database and attaching or linking additional information to Downloadables to be simple and straightforward. These features were well-known long before the time of the ‘926 patent (e.g., in the references discussed below). *See also supra* at ¶¶ 49-54.

## **VII. Construction of Certain Claim Terms**

### **A. “Database”**

89. It is my understanding that, in the Petition, Symantec has proposed that the broadest reasonable interpretation of the claim term “database” is: “an organized collection of data.” I agree with this construction. In my opinion, this is

consistent with the plain and ordinary meaning of the term “database,” as it would have been understood by one of ordinary skill in the art at the time of the ‘926 patent.

90. For example, Webster’s New World Dictionary of Computer Terms describes a database as a “coherent collection of data.” Ex. 1013, p. 95. Similarly, Webster’s Ninth New Collegiate Database describes a “database” as a “collection of data organized esp. for rapid search and retrieval (as by a computer).” Ex. 1012, p. 325. As another example, Webster’s College Dictionary describes a database as a “collection of organized, related data” Ex. 1011, p. 339. These contemporaneous definitions are consistent with the construction above.

91. Also, beyond the requirement in the claims that the database be indexed according to the Downloadable ID (e.g., the hash), the ‘926 patent does not attempt to define, limit, the operation of, or provide any particular structure for the claimed “database.” Instead, the patent and the claims themselves describe the types of data stored within the database, such as a Downloadable Security Profile (DSP) and how the database is indexed. ‘194 patent, col. 3:47-50 (“[t]he data storage device 230 stores a security database 240, which includes security information . . .”); col. 4:14-18; col. 9:52-55, FIGS. 2, 3; ‘926 patent, claim 15 (“retrieving security profile data for the incoming Downloadable from a database

of Downloadable security profiles indexed according to Downloadable IDs”). In my opinion, in view of this disclosure, one of ordinary skill in the art would have considered the “database” to be “an organized collection of data.”

## **VIII. Analysis of the Prior Art**

### **A. Anand**

92. Anand relates to “a system for downloading content from the Internet and controlling its actions on a client machine.” Anand, p. 1. Anand particularly recognized the benefits of distributing software (e.g., Java applets, plug-ins, ActiveX control) over the World Wide Web (WWW) and Internet. Anand, p. 1. Anand, however, believes there is a “significant concern with this approach is that the downloaded software may be malicious and may damage the user’s machine.” Anand, p. 1. To mitigate the effect of this malicious software, Anand attempts to “prevent content from (1) reading private files; (2) writing executable files; (3) limit access to their system’s CPU; and (4) prevent arbitrary remote communication from their system.” Anand, p. 1.


93. Anand teaches the use of content stamps to provide this protection. These content stamps are applied by a content rating service to content (e.g., software) provided by a manufacturer. Anand, p. 2-3, FIG. 1. Once “stamped,” the “manufacturer and content-rating services upload content and/or content

## **Conclusion**

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

I hereby 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 further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on the 10<sup>th</sup> day of September, 2015.

  
\_\_\_\_\_  
Jack W. Davidson

# **EXHIBIT 12**

*Inter Partes* Review of  
U.S. Patent No. 7,613,926

Filed on behalf of Symantec Corporation

UNITED STATES PATENT AND TRADEMARK OFFICE

---

BEFORE THE PATENT TRIAL  
AND APPEAL BOARD

---

SYMANTEC CORPORATION

Petitioner

v.

FINJAN, INC.

Patent Owner

---

Case To Be Assigned  
U.S. Patent No. 7,613,926

---

**DECLARATION OF JACK W. DAVIDSON IN SUPPORT OF  
PETITIONER PURSUANT TO 37 C.F.R. § 42.120**

Symantec 1019  
IPR of U.S. Pat. No. 7,613,926

96. In my opinion, it is clear that the ‘926 patent does not disclose anything new with respect to generating security profile data (*i.e.*, a list of potentially suspicious operations) for an executable. Instead the patent relies on well-known techniques for deriving such data, such as by parsing and decomposing executable code. ‘194 patent, col. 5:42-45. Thus, the only alleged differences in the challenged claims appears to be that a hash of the Downloadable is used to retrieve the DSP from a database and a representation of this DSP (as well as the Downloadable itself) is sent to the intended recipient. ‘926 patent, col. 21:58-22:4, claim 15. In my opinion one of ordinary skill in the art would have found these concepts, *i.e.*, using a hash to retrieve information from a database and attaching or linking additional information to Downloadables to be simple and straightforward. These features were well-known long before the time of the ‘926 patent (*e.g.*, in the references discussed below). *See also supra* at ¶¶ 49-54.

## **VIII. Construction of Certain Claim Terms**

### **A. “Database”**

97. It is my understanding that, in the Petition, Symantec has proposed that the broadest reasonable interpretation of the claim term “database” is: “an organized collection of data.” I agree with this construction. In my opinion, this is consistent with the plain and ordinary meaning of the term “database,” as it would



have been understood by one of ordinary skill in the art at the time of the '926 patent.

98. For example, Webster's New World Dictionary of Computer Terms describes a database as a "coherent collection of data." Ex. 1017, p. 95. Similarly, Webster's Ninth New Collegiate Database describes a "database" as a "collection of data organized esp. for rapid search and retrieval (as by a computer)." Ex. 1016, p. 325. As another example, Webster's College Dictionary describes a database as a "collection of organized, related data" Ex. 1015, p. 339. These contemporaneous definitions are consistent with the construction above.

99. Also, beyond the requirement that the database be indexed according to the Downloadable IDs, the '926 patent does not attempt to define, limit, the operation of, or provide any particular structure for the claimed "database." Instead, the patent and the claims themselves describe the types of data stored within the database, such as a Downloadable Security Profile (DSP) and how the database is indexed. '194 patent, col. 3:47-50 ("[t]he data storage device 230 stores a security database 240, which includes security information . . ."); col. 4:14-18; col. 9:52-55, FIGS. 2, 3; '926 patent, claim 15 ("retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs"). In my opinion, in view of this

disclosure, one of ordinary skill in the art would have considered the “database” to be “an organized collection of data.”

## **IX. Analysis of the Prior Art**

### **A. Touboul I**

#### **1. Touboul I teaches a computer-based method**

100. Touboul I discloses “a system and method for protecting a computer and a network from hostile Downloadables.” Touboul I, col. 1:23-27, claims 1, 66, 68. According to Touboul I, “this invention may be implemented using a programmed general purpose digital computer.” Touboul I, col. 9:58-10:1.

#### **2. Touboul I teaches a system for managing Downloadables**

101. Touboul I discloses “a system and method for protecting a computer and a network from hostile Downloadables.” Touboul I, col. 1:23-27, claims 32, 64. For this purpose, Touboul I, teaches an “internal network security system 110.” Touboul I, col. 2:66-3:61, FIGS. 1-3. By protecting computers from hostile Downloadables, Touboul I’s system “manages Downloadables.”

#### **3. Touboul I teaches a computer-readable storage medium storing program code**

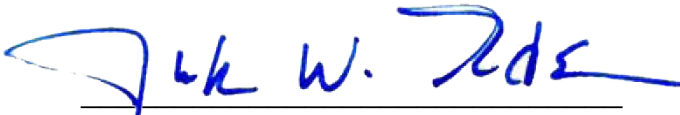
102. Touboul I teaches that “this invention may be implemented using a programmed general purpose digital computer.” Touboul I, col. 9:65-10:1, claim 65. For example, Touboul I teaches an “[i]nternal network security system 110

## **Conclusion**

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

I hereby 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 further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Executed on the 10<sup>th</sup> day of September, 2015.

  
\_\_\_\_\_  
Jack W. Davidson

**EXHIBIT 13**

---

U.S. Patent No. 8,141,154  
Declaration in Support of Petition for *Inter Partes* Review

**UNITED STATES PATENT AND TRADEMARK OFFICE**

---

**BEFORE THE PATENT TRIAL AND APPEAL BOARD**

---

Palo Alto Networks, Inc.  
Petitioner

v.

Finjan, Inc.  
Patent Owner

Patent No. 8,141,154  
Issue Date: Mar. 20, 2012

Title: System and Method for Inspecting Dynamically Generated Executable Code

---

*Inter Partes* Review No. IPR2016-00151

---

**DECLARATION OF DR. AVIEL D. RUBIN IN SUPPORT OF PETITION  
FOR *INTER PARTES* REVIEW**

**UNDER 35 U.S.C. §§ 311-319 AND 37 C.F.R. § 42.100 *et seq.***

**B. Broadest Reasonable Interpretation**

28. I have been informed and understand that patent claims are construed from the perspective of one of ordinary skill in the art at the time the claimed invention was made and that, during this proceeding, claims are to be given their broadest reasonable construction consistent with the specification.

29. I understand that, in *Inter Partes* Review, the claim terms are to be given their broadest reasonable interpretation (“BRI”) in light of the specification. *See* 37 C.F.R. §42.100(b). In performing my analysis and rendering my opinions, I have interpreted claim terms for which the Petitioner has not proposed a BRI construction by giving them the ordinary meaning they would have to the POSA reading the ’154 patent with its priority date, December 12, 2005, in mind, and in light of its specification and file history.

**C. Prior Art**

30. I have been informed and understand that a patent claim is invalid because of anticipation when every element of the claim is described in a single prior art reference, such that the elements are arranged as required by the claim. I have been informed and understand that the description of a claim element in a prior art reference can be express or inherent. For a prior art reference to describe a claim element inherently, the claim element must be necessarily present. Probabilities are not sufficient to establish inherency.

50. The dependent claims of the '154 patent also broadly recite features such as “suspending” and resuming of the second function,” “dynamically generated inputs,” and the invocation of “additional functions.” As will be shown further below, these broad recitations of features are readily taught or suggested by the references presented in the petition.

## **VI. CLAIM CONSTRUCTION**

### **1. “dynamically generated”**

51. The claim term “dynamically generated appears in dependent claims 3, 5, 8 and 11. Based on the claim language, the specification and the understanding of a person of ordinary skill in the art at the time of the alleged invention of the '154 patent, the broadest reasonable interpretation of the term “dynamically generate[d]” is: “generate[d] at run-time.”

52. Claims 3, 5, 8, and 11 each recite that the input associated with the first function is “dynamically generate[d].” These dependent claims make clear that the input is generated while the content processor is processing the content and invoking the functions (*i.e.*, during run-time).

53. The proposed construction is also consistent with the specification of the '154 patent. The '154 patent is replete with disclosure equating dynamically generated inputs to inputs that are generated at run-time. In one example, the '154 patent explains that “viruses take advantage of features of dynamic HTML

generation . . . to generate themselves on the fly at runtime.” (’154 patent, 3:34-37.) In another example, the ’154 patent states that “[s]ince the input to the function is being passed at run-time, it has already been dynamically generated.” (’154 patent, 4:44-45, 4:21-33, 4:50-53.) Thus, one of ordinary skill in the art at the time of the ’154 patent would have understood the term “dynamically generate[d],” as used in the ’154 patent, to mean “generate[d] at run-time.” This construction of the term dynamically generated comports with the broadest reasonable interpretation that a person of ordinary skill in the art would have of the term given the disclosure in the ’154 patent.

## **VII. STATE OF THE ART**

### **1. Viruses Scanning and Heuristics**

54. A computer program is a sequence of instructions that tell a computer processor what to do. Although the author of a computer program would be understandably upset if processors refused to obey the instructions, the blind obedience they are designed to deliver makes them incredibly vulnerable to misuse. Computer processors will not question the instructions they are given no matter how harmful. And processors can run millions and billions of instructions per second all day and all night whether a human is monitoring them or not.

55. Even when they can be monitored, computer instructions in a software program are generally too large, too complicated, and too cryptic for a human to



## **XI. CONCLUSION**

169. For at least the preceding reasons, I believe that each of claims 1-12 of the '154 patent is invalid pursuant to 35 U.S.C. § 103.

170. I may testify about any of the preceding topics at a deposition or hearing.

171. I reserve the right to respond to any declarations that are submitted by Finjan's expert witnesses or to any testimony by Finjan's fact or expert witnesses, whether at deposition or at trial.



---

Aviel D. Rubin, Ph.D.