No. 21-2093

IN THE

United States Court of Appeals for the Federal Circuit

FINJAN LLC

Plaintiff-Appellant

v.

ESET, LLC, ESET SPOL. S.R.O.,

Defendants-Appellees.

Appeal from the United States District Court
for the Southern District of California,
No. 3:17-cv-00183-CAB-BGS

OPENING BRIEF OF FINJAN LLC

Juanita R. Brooks
FISH & RICHARDSON P.C.
12860 El Camino Real, Ste 400
San Diego, CA 92130
Telephone: 858-678-5070

Robert Courtney
FISH & RICHARDSON P.C.
60 S 6th St., Suite 3200
Minneapolis, MN 55402
Telephone: 612-335-5070

August 23, 2021

Attorneys for Plaintiff-Appellant

Claim 1 of U.S. Patent No. 6,154,844 (Appx169–185)

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.

Claim 9 of U.S. Patent No. 6,804,780 (Appx187–204)

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.

Claim 1 of U.S. Patent No. 8,079,086 (Appx240–264)

1. A computer-based method, comprising the steps of:

receiving an incoming ***Downloadable***;

deriving security profile data for the ***Downloadable***, including a list of suspicious computer operations that may be attempted by the ***Downloadable***;

appending a representation of the Downloadable security profile data to the ***Downloadable***, to generate an appended Downloadable; and

transmitting the appended Downloadable to a destination computer.

Claim 1 of U.S. Patent No. 9,189,621 (Appx266–300)

1. A system for determining whether a ***downloadable*** is suspicious, comprising:

at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:

a plurality of operating system probes operating substantially in parallel for monitoring a plurality of subsystems of the operating system during runtime for an event caused from a request made by a ***downloadable***;

an interrupter for interrupting processing of the request;
a first comparator coupled to the plurality of operating system probes for comparing information pertaining to the ***downloadable*** against a predetermined security policy, wherein the information pertaining to the ***downloadable*** includes information pertaining to an operation of the ***downloadable*** and distinct from information pertaining to the request; and
a response engine for performing a predetermined responsive action based on the comparison.

Claim 3 of U.S. Patent No. 9,219,755 (Appx302–338)

3. A system for reviewing an operating system call issued by a ***downloadable***, comprising:
at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:
a downloadable engine for intercepting a request message being issued by a ***downloadable*** to an operating system, wherein the request message includes an extension call;
a request broker for receiving a notification message from the downloadable engine regarding the extension call;
a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine, intercepting an operating system call being issued by the ***downloadable*** to an operating system and associated with the operating system function and providing an event message regarding the operating system call;
a runtime environment monitor for receiving the notification message and the event message and comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and
a response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

CERTIFICATE OF INTEREST

Counsel for Plaintiff-Appellant Finjan LLC ("Finjan") certifies the following:

1. Provide the full names of all entities represented by undersigned counsel in this case.

   Finjan LLC

2. Provide the full names of all real parties in interest for the entities. Do not list the real parties if they are the same as the entities.

   Finjan LLC

3. Provide the full names of all parent corporations for the entities and all publicly held companies that own 10% or more stock in the entities.

   Finjan Holdings LLC, CFIP Goldfish Holdings LLC

4. List all law firms, partners, and associates that (a) appeared for the entities in the originating court or agency or (b) are expected to appear in this court for the entities. Do not include those who have already entered an appearance in this court. Fed. Cir. R. 47.4(a)(4).

   Kramer Levin Naftalis & Frankel: Benu C. Wells, Matthew W. Olinzock, Aaron M. Frankel, Cristina L. Martinez, James R. Hannah, Linjun Xu, Michael H. Lee, Phuong D. Nguyen, Yuridia Caire

   Kramer Levin LLP: Paul J. Andre, Gregory C. Proctor, Hannah Y. Lee, Hien Khanh Lien, Kristopher B. Kastens, Lisa Kobialka

   Fish & Richardson P.C.: Lawrence R. Jarvis, Jason W. Wolff, K. Nicole Williams, Michael Amon, Roger A. Denning

5. Provide the case titles and numbers of any case known to be pending in this court or any other court or agency that will directly affect or be directly affected by this court's decision in the pending appeal. Do not include the originating case number(s) for this case. Fed. Cir. R. 47.4(a)(5). See also Fed. Cir. R. 47.5(b).

***Finjan LLC v. Qualys Inc.*, No. 4:18-cv-07229 (N.D. Cal.)**

***Finjan, Inc. v. SonicWall, Inc.*, No. 5:17-cv-04467 (N.D. Cal.)**

***Finjan, Inc. v. Rapid7, Inc.*, No. 1:18-cv-01519 (D. Del.)**

***Finjan LLC v. Palo Alto Networks, Inc.*, No. 4:14-cv-4908 (N.D. Cal.)**

***Finjan, Inc. v. Fortinet, Inc.*, No. 3:18-cv-06555 (N.D. Cal.)**

6. Provide any information required under Fed. R. App. P. 26.1(b) (organizational victims in criminal cases) and 26.1(c) (bankruptcy case debtors and trustees). Fed. Cir. R. 47.4(a)(6).

Not applicable

August 23, 2021 */s/ Juanita R. Brooks*
Juanita R. Brooks

TABLE OF CONTENTS

Statement of Related Cases 1

Statement of Jurisdiction 1

Statement of the Issues 2

Statement of the Case 2

A. Finjan and its Network Security Innovations 2

B. Finjan's Complaint Against ESET 3

C. This Appeal's Patents, Which All Claim "Downloadables" 3

D. The Record Evidence Regarding Interpretation of "Downloadable" 9

1. The Unasserted Patents Relevant to "Downloadable" 10

2. "Downloadable" in the '639 Provisional 11

3. "Downloadable" in the '520 and '962 Unasserted Patents 12

4. "Downloadable" in the '844 and '780 Patents 13

5. The Incorporations by Reference 14

E. The Case Below 17

1. The District Court's Claim Construction Order 17

2. The District Court's Denial of ESET's First Indefiniteness Motion 18

3. Mistrial Due to COVID-19 19

4. The District Court's Summary Judgment of Indefiniteness 20

5. Denial of Reconsideration, Final Judgment, and Appeal 22

Summary of the Argument 24

Argument 26

I. STANDARD OF REVIEW 26

II. THE DISTRICT COURT ERRED WHEN IT DETERMINED THAT THE CLAIMED DOWNLOADABLE MUST BE "SMALL," AND REVERSAL OF THE INDEFINITENESS ORDER IS WARRANTED 27

A. The '520 and '962 Unasserted Patents Have a Unique Descriptive Focus on "Applets" 28

B. Incorporation by Reference Does Not Make "Small" a Requirement of "Downloadables" for All Asserted Patents 31

1. The Later Filed '844 and '780 Patents Expressly Restate a Broader Definition of "Downloadable," and Nothing in their Context Suggests Adoption of the "Small" Restriction 34

2. The '086 Patent 35

3. The '621 and '755 Patents 37

C. Upon Correcting the District Court's Claim Interpretation, the Court Should Reverse the Indefiniteness Order, Vacate the Judgment, and Remand 38

III. SHOULD THE COURT DETERMINE THAT THE CLAIMED DOWNLOADABLE MUST BE "SMALL" THE DISTRICT COURT STILL ERRED WHEN IT GRANTED SUMMARY JUDGMENT TO ESET, AND REVERSAL OF THE INDEFINITENESS ORDER IS WARRANTED 39

A. The District Court Reversibly Erred When It Placed the Burden of Establishing the Definiteness of "Small" on Finjan Without First Requiring ESET to Come Forward with Evidence of Skilled Artisans' Inability to Understand "Small" 40

B. The District Court Reversibly Erred When It Granted Summary Judgment Based on a Facially Incomplete Trial Record 43

C. The District Court Reversibly Erred When It Determined That, on the Available Record, ESET had Proved Finjan's Claims Indefinite 45

1. The Intrinsic Evidence and Truncated Trial Record Alone Demonstrate Why Summary Judgment Should Have Been Denied 47

2. Opinion Testimony That Finjan Never Had a Chance to Present at Trial Further Confirms Definiteness 48

D. The Court Should Reverse the Indefiniteness Order, Vacate the Judgment, and Remand 51

Conclusion 52

ADDENDUM 53

CERTIFICATE OF SERVICE AND FILING i

CERTIFICATE OF COMPLIANCE ii

TABLE OF AUTHORITIES

Page(s)

Cases

Asics America Corp. v. Akeva L.L.C.,
No. 1:09-cv-135, 2015 WL 1013645 (M.D.N.C. Mar. 9, 2015) 33

Biosig Instruments, Inc. v. Nautilus, Inc.,
783 F.3d 1374 (Fed. Cir. 2015) [*Nautilus II*] 46

Blue Coat Sys. LLC v. Finjan, Inc.,
No. IPR2017-00997, 2017 WL 4349402 (PTAB Sept. 5, 2017) 4

Cisco Sys., Inc. v. Finjan, Inc.,
No. IPR2017-02154, 2018 WL 1635926 (PTAB Apr. 3, 2018) 4

ESET, LLC v. Finjan, Inc.,
No. IPR2017-01969, 2018 WL 340652 (PTAB Jan. 9, 2018) 4

Finjan LLC v. Palo Alto Networks, Inc.,
No. 4:14-cv-04908 (N.D. Cal.) 1

Finjan LLC v. Qualys Inc.,
No. 4:18-cv-07229 (N.D. Cal.) 1, 21, 22

Finjan, Inc. v. Fortinet, Inc.,
No. 3:18-cv-06555 (N.D. Cal.) 1

Finjan, Inc. v. Rapid7, Inc.,
No. 1:18-cv-01519 (D. Del.) 1

Finjan, Inc. v. SonicWall, Inc.,
No. 5:17-cv-04467 (N.D. Cal.) 1

Juniper Networks, Inc. v. Finjan, Inc.,
No. IPR2019-00026, 2020 WL 1696139 (PTAB Apr. 7, 2020) 4

Modine Mfg. Co. v. U.S. Int'l Trade Comm'n,
75 F.3d 1545 (Fed. Cir. 1996).....31

MPHJ Tech. Invs., LLC v. Ricoh Ams. Corp.,
847 F.3d 1363 (Fed. Cir. 2017) 32

Nautilus, Inc. v. Biosig Instruments, Inc.,
572 U.S. 898 (2014) [*Nautilus I*] 40, 46

Nevro Corp. v. Boston Sci. Corp.,
955 F.3d 35 (Fed. Cir. 2020) 26, 27

One-E-Way, Inc. v. Int'l Trade Comm'n,
859 F.3d 1059 (Fed. Cir. 2017) 41

Paice LLC v. Ford Motor Co.,
881 F.3d 894 (Fed. Cir. 2018) 31

Palo Alto Networks, Inc. v. Finjan, Inc.,
No. IPR2016-00145, slip op. (PTAB May 9, 2016) 4

Palo Alto Networks, Inc. v. Finjan, Inc.,
No. IPR2016-00165, slip op. (PTAB Apr. 21, 2016) 4

Presidio Components, Inc. v. Am. Tech. Ceramics Corp.,
875 F.3d 1369 (Fed. Cir. 2017) 43, 44

Rustico v. Intuitive Surgical, Inc.,
993 F.3d 1085 (9th Cir. 2021) 26

Smartphone Technologies LLC v. Research in Motion Corp.,
No. 6:10-cv-74, 2012 WL 489112 (E.D. Tex. Feb. 13, 2012) 32, 33

Sonix Tech. Co. v. Publications Int'l Ltd.,
844 F.3d 1370 (Fed. Cir. 2017) 46, 48

Symantec Corp. v. Finjan, Inc.,
No. IPR2015-01894, 2016 WL 1082326 (PTAB Mar. 11, 2016) 4

Tech. Lic'g Corp. v. Videotek, Inc.,
545 F.3d 1316 (Fed. Cir. 2008) 41

Unified Patents, Inc. v. Finjan, Inc.,
No. IPR2019-01611, 2020 WL 1855822 (PTAB Apr. 13, 2020) 4

X2Y Attenuators, LLC v. International Trade Commission,
757 F.3d 1358 (Fed. Cir. 2014) 32, 33, 35

Young Dental Mfg. Co. v. Q3 Special Prods., Inc.,
112 F.3d 1137 (Fed. Cir. 1997) 32

Statutes

28 U.S.C. § 41 26

28 U.S.C. § 1295(a)(1) 1

28 U.S.C. § 1338(a) 1

28 U.S.C. § 2107(a) 1

35 U.S.C. § 282 ..... 40, 43

STATEMENT OF RELATED CASES

Per Federal Circuit Rule 47.5(a), Finjan states that no other appeals from this civil action have previously been before this or any other appellate court.

Per Federal Circuit Rule 47.5(b), the undersigned counsel is aware of the following pending cases that will directly affect or be directly affected by this Court's decision in the pending appeal:

- *Finjan LLC v. Qualys Inc.*, No. 4:18-cv-07229 (N.D. Cal.)
- *Finjan, Inc. v. SonicWall, Inc.*, No. 5:17-cv-04467 (N.D. Cal.)
- *Finjan, Inc. v. Rapid7, Inc.*, No. 1:18-cv-01519 (D. Del.)
- *Finjan LLC v. Palo Alto Networks, Inc.*, No. 4:14-cv-04908 (N.D. Cal.)
- *Finjan, Inc. v. Fortinet, Inc.*, No. 3:18-cv-06555 (N.D. Cal.)

STATEMENT OF JURISDICTION

This is an appeal from a patent case and the district court had subject-matter jurisdiction under 28 U.S.C. § 1338(a). This Court has appellate jurisdiction under 28 U.S.C. § 1295(a)(1). Finjan represents that the judgment appealed from [Appx25] is final and disposes of all parties' claims. Finjan filed a notice of appeal on June 17, 2021 [Appx23541–23544], timely per 28 U.S.C. § 2107(a).

STATEMENT OF THE ISSUES

1. Did the district court err as a matter of law when it determined that the term "Downloadable" in all asserted claims was limited to a "small" program?

2. If the claimed "Downloadable" must be "small," did the district court err as a matter of law when it determined "small" to be indefinite, thus rendering claims requiring a "Downloadable" invalid?

STATEMENT OF THE CASE

A. Finjan and its Network Security Innovations

Finjan, LLC ("Finjan") developed inventive solutions regarding dynamic threat detection for computer networks. Founded in Israel in 1996, Finjan innovated in adaptive threat detection for computer networks. Finjan's technology allows a company's cybersecurity platform to identify and block even never-before-detected viruses or other malware attempting to maliciously penetrate the network.

The earliest roots of Finjan's innovation go back to the work of Shlomo Touboul (the company's founder and co-inventor on all appealed patents) in ***behavior analysis*** for network threat detection. One concept at play in this appeal is Finjan's technology for monitoring, and learning from, the behavior of "Downloadables," i.e., programs downloaded from the Internet (or from

some other location) that might contain security threats. With Finjan's technology, it is possible to draw conclusions and make assumptions about Downloadables that a system has never actually seen before, using lessons from similar Downloadables observed in the past.

B. Finjan's Complaint Against ESET

ESET is a network computer security company that sells various security products for the "cloud," for servers, and for endpoints, each of which uses Finjan's patented technology. In 2016, Finjan filed suit against ESET in the Northern District of California, seeking relief under six patents. Five of those, all of which are part of the same family, are part of the present appeal.[1]

C. This Appeal's Patents, Which All Claim "Downloadables"

This appeal concerns five Finjan patents linked by shared ancestry:

Patent No.	***Issue Date***	***Inventors***	***Citation***
6,154,844 ("the '844 Patent")	Nov. 28, 2000	S. Touboul & N. Gal	Appx169–185
6,804,780 ("the '780 Patent")	Oct. 12, 2004	S. Touboul	Appx187–204
8,079,086 ("the '086 Patent")	Dec. 13, 2011	Y. Edery et al.	Appx240–264
9,189,621 ("the '621 Patent")	Nov. 17, 2015	S. Touboul	Appx266–300
9,219,755 ("the '755 Patent")	Dec. 22, 2015	S. Touboul	Appx302–338

[1] On a joint request by the parties, the district court severed and stayed Finjan's claims under the sixth patent pending resolution of the claims at issue in this appeal. *See* Appx23469.

The five patents each address different aspects of secure computer systems and networks against malware and other security threats. They include patents that have repeatedly survived *inter partes* review scrutiny, and whose claims have been interpreted and adjudicated across numerous proceedings.[2] Relevant to this appeal, all five describe interaction between a security system and a "Downloadable." The interpretation of "Downloadable" is central to this appeal (and disputed by the parties), but the core concept is a computer file that (1) is downloaded from some source (e.g., from the Internet), and (2) contains code that can be run or interpreted by a destination computer. *See, e.g.*, Appx178 (1:44–47). Each of Finjan's asserted

[2] *See, e.g.*, *Unified Patents, Inc. v. Finjan, Inc.*, No. IPR2019-01611, 2020 WL 1855822 (PTAB Apr. 13, 2020) (denying IPR institution on '086 Patent); *Juniper Networks, Inc. v. Finjan, Inc.*, No. IPR2019-00026, 2020 WL 1696139 (PTAB Apr. 7, 2020) (confirming patentability for '844 Patent); *Cisco Sys., Inc. v. Finjan, Inc.*, No. IPR2017-02154, 2018 WL 1635926 (PTAB Apr. 3, 2018) (denying IPR institution on '844 Patent); *ESET, LLC v. Finjan, Inc.*, No. IPR2017-01969, 2018 WL 340652 (PTAB Jan. 9, 2018) (denying IPR institution on '086 Patent); *Blue Coat Sys. LLC v. Finjan, Inc.*, No. IPR2017-00997, 2017 WL 4349402 (PTAB Sept. 5, 2017) (denying IPR institution on '755 Patent); *Palo Alto Networks, Inc. v. Finjan, Inc.*, No. IPR2016-00145, slip op. (PTAB May 9, 2016), Paper #9 [Appx2417–2425] (denying IPR institution on '926 Patent); *Palo Alto Networks, Inc. v. Finjan, Inc.*, No. IPR2016-00165, slip op. (PTAB Apr. 21, 2016), Paper #9 [Appx1453–1474] (denying IPR institution on '780 Patent); *Symantec Corp. v. Finjan, Inc.*, No. IPR2015-01894, 2016 WL 1082326 (PTAB Mar. 11, 2016) (denying IPR institution on '844 Patent).

patents teaches and claims a security system analyzing Downloadables, and interacting with the computers that download or run them, so as to provide enhanced computer security. The precise security advantages, and the claims, vary by patent.

In the '844 Patent, a security system inspects a Downloadable, generates a "security profile" for it, and then links the profile to the Downloadable so that in the future decisions can be made about whether to permit or block it. Appx178 (1:62–2:2). This security profile "preferably includes a list of all potentially hostile or suspicious computer operations that may be attempted by the Downloadable[.]" Appx179 (4:4–7). Claim 1 of the '844 patent is representative:

> **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.

Appx183 (11:13–21).

In the '780 Patent, a security system assembles information about whatever additional components the Downloadable may attempt to execute when run. Appx199 (4:50–66). Those additional components might contain

security threats not present in the Downloadable itself. The system uses a "hashing function" to cryptographically combine information about the Downloadable and about the components, forming a "Downloadable ID." Appx200 (4:54–56). As a result, in the future, other security programs can use the ID to quickly reference security information. Claim 9 is representative here:

> **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.

Appx203 (10:49–59).

In the '086 Patent, "security profile data" is derived for the Downloadable, indicating suspicious operations that the Downloadable might attempt. Appx257 (9:66–10:12). This is similar to the "security profile" of the '844 Patent. In the '086 Patent, however, information is appended to the Downloadable itself that will represent this security profile. Appx258 (11:16–26). Claim 1 is representative:

> **1.** A computer-based method, comprising the steps of:
> receiving an incoming Downloadable;
> deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable;
> appending a representation of the Downloadable security profile data to the Downloadable, to generate an appended Downloadable; and
> transmitting the appended Downloadable to a destination computer.

Appx262 (20:60–21:2).

In the '621 Patent, the focus is on security software installed within a client computer. It describes a system that can compare a Downloadable, when it encounters one, to a security policy it previously received, and on that basis determine if the Downloadable is safe to run. Claim 1 is representative:

> 1. A system for determining whether a downloadable is suspicious, comprising:
> at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:
> a plurality of operating system probes operating substantially in parallel for monitoring a plurality of subsystems of the operating system during runtime for an event caused from a request made by a downloadable;
> an interrupter for interrupting processing of the request;
> a first comparator coupled to the plurality of operating system probes for comparing information pertaining to the downloadable against a predetermined security policy, wherein the information pertaining to the downloadable

> includes information pertaining to an operation of the downloadable and distinct from information pertaining to the request; and
> a response engine for performing a predetermined responsive action based on the comparison.

Appx300 (21:33–53).

In the '755 Patent, the focus is again on protecting a client computer. Here, the client computer has programmatic "probes" in its core systems that can detect if a problematic Downloadable is about to be run, and can intercede to make sure security is not breached. Claim 3 is representative:

> 3. A system for reviewing an operating system call issued by a downloadable, comprising:
> at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:
> a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;
> a request broker for receiving a notification message from the downloadable engine regarding the extension call;
> a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine, intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function and providing an event message regarding the operating system call;
> a runtime environment monitor for receiving the notification message and the event message and

> comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and
>
> a response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

Appx337 (22:30–62).

D. The Record Evidence Regarding Interpretation of "Downloadable"

This appeal focuses on the construction of "Downloadable," as used in all five asserted patents. Because the five asserted patents are part of a single family, there is overlap among their specifications, but also substantial divergence. As discussed *infra*, the patents have varying focus, which affects the interpretation of each patent's specific claims.

Each of the asserted patents incorporates by reference other applications within the same family. Some of these applications matured into asserted patents; others matured into patents not asserted here. The following section identifies the various patent documents of interest, beginning with documents from unasserted patents.

1. The Unasserted Patents Relevant to "Downloadable"

Several patent documents are part of the intrinsic record for "Downloadable" that are neither asserted patents nor part of those patents' direct prosecution history. One is a provisional application. The other three are unasserted patents. All four are potentially relevant to the meaning of "Downloadable" in the asserted patents, due to incorporation by reference:[3]

- **The '639 Provisional** is U.S. Provisional Patent Application No. 60/030,639 (filed Nov. 8, 1996). Appx1859–1891. The '639 Provisional is the first patent filing in the family containing all the asserted patents, and the unasserted patents described below.

- **The '520 Unasserted Patent** is U.S. Patent No. 6,167,520 (filed Jan. 29, 1997). Appx1008–1022. The '844 and '780 Patents incorporate the'520 Unasserted Patent by reference. *See* Appx178 (1:15) (incorp'g U.S. Patent App. 08/790,097); Appx199 (1:20) (same).

- **The '962 Unasserted Patent** is U.S. Patent No. 6,480,962 (filed Apr. 18, 2000). Appx1038–1054. It is a continuation of the '520 Unasserted Patent, and its written description is substantially the same. The '086, '621, and '755 Patents incorporate the '962 Unasserted Patent by reference. *See* Appx253(1:31–35); Appx290(1:54–59); Appx327(1:54–59); and

- **The '194 Unasserted Patent** is U.S. Patent No. 6,092,194 (filed Nov. 6, 1997). Appx984–1006. The '844 and '086 Patents incorporate the '194 Unasserted Patent by reference. *See* Appx178 (1:12) (incorp'g U.S. Patent App. 08/964,388); Appx253 (1:26–27) (same).

[3] A chart prepared by the district court diagramming the relevant incorporations by reference is at Appx13.

2. "Downloadable" in the '639 Provisional

"Downloadable" was first defined in the '639 Provisional:[4]

> [S]ecurity systems are not configured to recognize computer viruses which have been attached to or configured as Downloadable application programs, commonly referred to as "applets" or "Downloadables." ***A Downloadable is an executable application program which is automatically downloaded from a source computer and run on the destination computer.*** Examples of Downloadables include applets designed for use in the Java™ distributing environment produced by Sun Microsystems or for use in the Active X distributing environment produced by Microsoft Corporation. Therefore, a system and method are needed to protect computers from viruses attached to these Downloadables.

Appx1863 at 2 (emphasis added); *see also* Appx178 (1:44–47) (restating emphasized portion); Appx199 (1:50–54) (same).

The remaining disclosure of the '639 Provisional makes no reference to Downloadables being "small," nor does it suggest that applets or other small Downloadables have any particular prominence to the invention(s) being described.

[4] As discussed *infra*, the emphasized portion of the '639 Provisional's "Downloadable" definition below was restated in the '844 and '780 Patents.

3. "Downloadable" in the '520 and '962 Unasserted Patents

As to "Downloadable," the '520 Unasserted Patent, and its continuation the '962 Unasserted Patent, differ from the '639 Provisional. The '520 Unasserted Patent was filed two months after the '639 Provisional. *Compare* Appx1859 ('639 Provisional filed Nov. 8, 1996) *with* Appx1008 ('520 Unasserted Patent filed Jan. 29, 1997). Like the Provisional, the '520 Unasserted Patent includes a definition of "Downloadable." But neither the "Downloadable" definition, nor the language introducing it, are the same. Notably for this appeal, the '520 Unasserted Patent drew particular attention to the "applet" variety of Downloadable, and defined the term "Downloadable" as requiring a "small" application program:

> [S]ecurity systems are typically not configured to recognize computer viruses which have been attached to or masked as harmless Downloadables (i.e., ***applets***). A Downloadable is a ***small*** executable or interpretable application program which is downloaded from a source computer and run on a destination computer.

Appx1016 (1:32–34 (emphasis added)); *see also* Appx1047 (1:36–41 (same)).

Unlike the '639 Provisional (and unlike the asserted patents), the '520 and '962 Unasserted Patents have a particular focus on "applets." In addition to the definition immediately above, the '520 and '962 Unasserted Patents add

further description about particular "security holes in Downloadable interpreters" that parse and run applets. Appx1016 (1:39–51); Appx1047 (1:46–58 (same)). This focus on "interpreters" and their applets was not in the '639 Provisional, and does not appear in the other patents.

The focus on "applets" continues in the detailed description in the '520 and '962 Unasserted Patents. Figures 3 and 4 depict security systems particularly encompassing technology for running applets in the form of Java applets (Figure 3) and ActiveX controls (Figure 4). Again, no such disclosure is in the other patents. Appx1011, Appx1017 (fig.3, 3:42–49); Appx1042, Appx1048 (fig.3, 3:52–63 (same)).

4. "Downloadable" in the '844 and '780 Patents

The '844 Patent, the '194 Unasserted Patent, and the '780 Patent—all filed after the'520 Unasserted Patent—repeat verbatim the definition of "Downloadables" from the '639 Provisional, though they remove the reference to "applets" from the preceding sentence. Appx178 (1:44–47); Appx996 (1:44–47); Appx199 (1:50–53). As can be seen in the quote below, there is no indication that the Downloadable contemplated here must be "small":

> [S]ecurity systems are not configured to recognize viruses which have been attached to or configured as "Downloadables." **A Downloadable is an executable application program, which is downloaded from a**

> **source computed and run on the destination computer**.

Appx178 (1:44–47 (emphasis added)).

Additionally, the ’844 and ’780 Patents lack the tight focus on “applets” observed in the ’520 and ’962 Unasserted Patents. Figure 5 notably depicts a “generic protection engine 500,” which includes—but is not limited to—various security techniques from the ’520 Unasserted Patent. Appx174, Appx181 (fig.5, 7:41–8:16 (describing figure 5), 8:17–36 (describing how the engine may “include” components from incorporated applications)); Appx193, Appx202 (fig.5, 7:20–59 (same)). And none of the figures or descriptions of the ’844 and ’780 patents are tightly focused on Java “applets,” as is the description in the ’520 and ’962 Unasserted Patents.

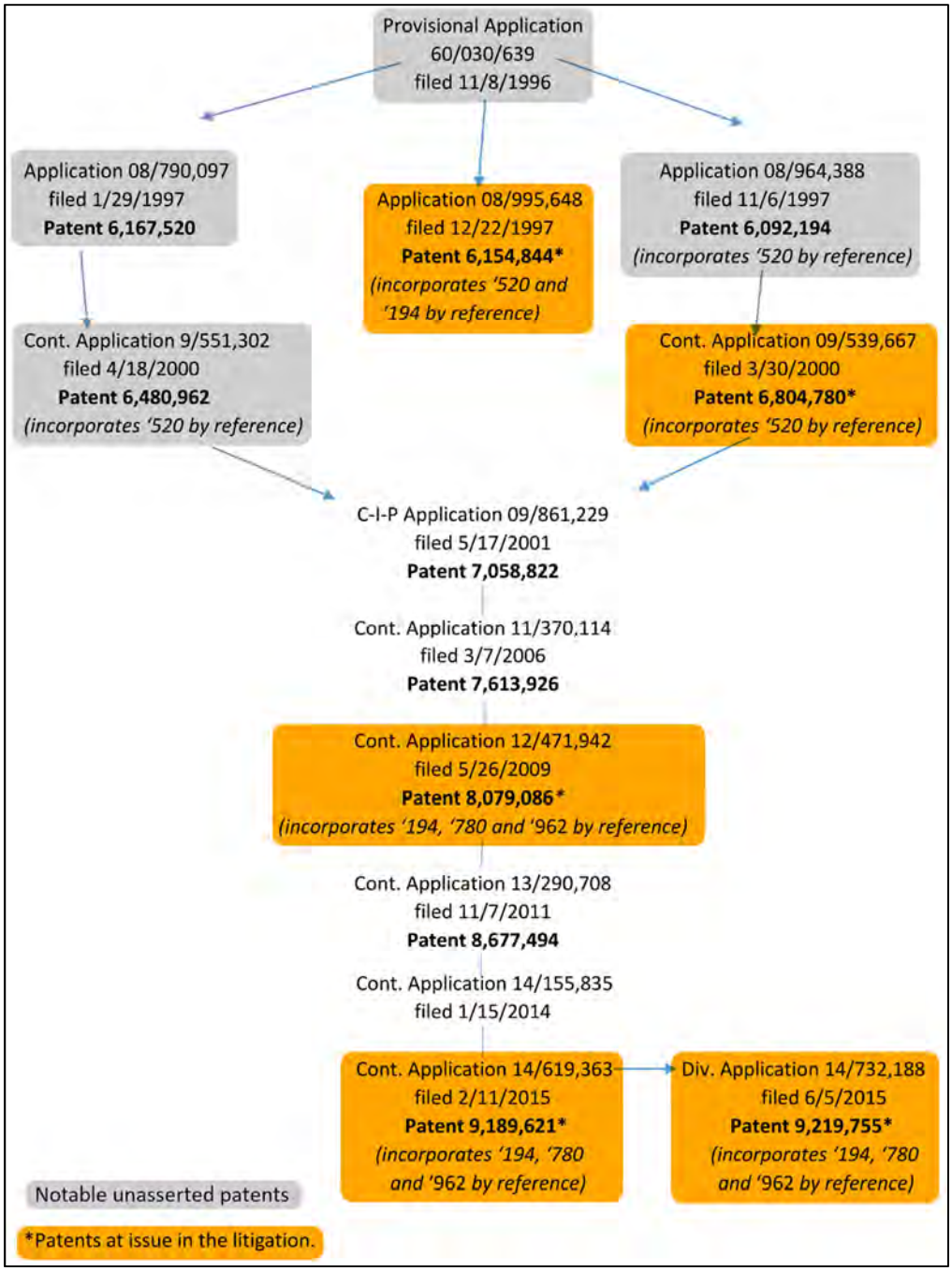
5. The Incorporations by Reference

All five asserted patents employ various incorporations by reference that reach back to one or both of the above definitions, as follows:

- As already noted, **the ’844 and ’780 Patents** each ***restate*** the definition of “Downloadable” from the ’639 Provisional. Appx178 (1:44–47); Appx199 (1:50–53). They also **incorporate** by reference:
  - The ’520 Unasserted Patent (*see* Appx178 (1:15–19); Appx199 (1:19–23)); and
  - The ’194 Unasserted Patent (*see* Appx178 (1:10–14); Appx199 (1:7–12)).

- **The ’086, ’621, and ’755 Patents** each **incorporate** the following by reference:
  - The ’194 Unasserted Patent (*see* Appx253 (1:25–30)); Appx290 (1:43–48); Appx327 (1:45–49));
  - The ’780 Patent (*see* Appx253 (1:19–25); Appx290 (1:37–41)); Appx327 (1:40–44)); and
  - The ’962 Unasserted Patent (*see* Appx253 (1:30–35); Appx290 (1:54–59); Appx327 (1:55–59)).

The district court attached the following diagram to its claim construction order, highlighting the incorporations by reference among the asserted patents:



Appx13 (highlighting, explanatory text added).

E. The Case Below

Finjan filed the present case against ESET in July 2016. Appx131–132, Appx165–167. The case proceeded to claim construction, with Finjan asserting the following claims:

Asserted Patent	***Asserted Claims***
'844 Patent	1, 7, 11, 15, 16
'780 Patent	9, 13, 18
'086 Patent	1, 2, 9, 10, 24
'621 Patent	1, 5, 10, 13, 15
'755 Patent	3, 6

Appx767.

1. The District Court's Claim Construction Order

The district court entered its claim construction order on November 14, 2017. Appx1–11. Relying primarily on the definition of "Downloadable" in the earlier-filed '520 Unasserted Patent, the Court held that "Downloadable" should be construed, for all asserted patents, to mean:

> "a small executable or interpretable application program which is downloaded from a source computer and run on a destination computer"

Appx3.

The district court derived this definition from the '520 and '962 Unasserted Patents. Appx2–3. No party had proposed applying this definition to all asserted patents.[5]

The district court reasoned that, with the various incorporations by reference, there were "two branches of the family tree" in play for the asserted patents. Appx2–3. The court stated that the definition from the '520 and '962 Unasserted Patents (which required "small" Downloadables) was generally supported by the specifications of the asserted patents, though the court did not describe that support, nor why the definitions of "Downloadable" lacking the "small" restriction were not equally supported. Appx3.

2. The District Court's Denial of ESET's First Indefiniteness Motion

The case proceeded. After discovery closed, ESET filed a motion for summary judgment alleging that the term "Downloadable," as it had been construed, was indefinite. *See generally* Appx5510 (alleging, "'small' is a vague term of degree."). Finjan opposed, pointing out that both sides' experts had successfully interpreted and applied the term, and noting that this was a

[5] ESET had proposed it for the '621 and '755 Patents, but not the others. *See generally* Appx796, Appx805.

new theory, not previously ventured by ESET. *See generally* Appx20066–20069.

The district court denied ESET's motion without prejudice to revisit at trial. Appx22637–22639. The court noted a "conflicting record" among the experts as to the interpretation of "small," and so held that ESET had not demonstrated indefiniteness by clear and convincing evidence. Appx22638–22639.

3. Mistrial Due to COVID-19

Trial began on Wednesday, March 10, 2020. Appx22815–22837. Opening arguments were presented, and Finjan began its case-in-chief, with intermittent cross-examination by ESET. Appx22860–22861. By Friday afternoon (March 12), Finjan had presented several witnesses, but only one of its liability experts (computer scientist Eric Cole, Ph.D., discussing the '844 Patent only) had given full testimony. *See* Appx23019–23020, Appx23215. The district court adjourned for the weekend, noting that that the public health situation concerning COVID-19 was "developing rapidly":

> THE COURT: [W]e will recess for the day and for the rest of the week now. So as you [the jury] leave for the weekend, just remember not to discuss the case with anyone, not to do any research on your own over the course of the weekend. Be here Monday morning at 8:30 when we will continue. Things in the world are developing rapidly. I have every expectation we will

> all be here Monday morning and that the court will be open. However, should things in your lives change, please let us know as soon as possible. And if anything happens with the court, obviously, we will be in touch with you.
>
> My expectation is that we'll be here. And I know they're taking extra steps after groups have been through to make sure that everything is wiped down and cleaned off here in the courtroom. But, you know, who knows? So everybody be safe, take care of yourselves, and we'll see you on Monday.

Appx23313 (563:2–16).

The COVID-19 situation continued to escalate. Trial did not resume on Monday, March 15. On the following morning, March 16, the district court excused the jury, suspended proceedings, and with the parties' agreement declared a mistrial.[6] Appx121 (ECF #783).

4. The District Court's Summary Judgment of Indefiniteness

Through the first half of 2020, the district court and the parties continued to advance various aspects of the case via briefing and remote appearances. Then, in August 2020, ESET "renewed" its motion on indefiniteness. Appx23362. ESET argued that, though the trial had ended

[6] The following day, Chief Judge Burns ordered the continuation of all civil jury trials in the district. Order of the Chief Judge No. 18 (S.D. Cal. Mar. 17, 2020), *available at* https://www.casd.uscourts.gov/_assets/pdf/rules/Order%20of-%20the%20Chief%20Judge%2018.pdf (accessed Aug. 3, 2021).

prematurely, the record was nonetheless ready for an indefiniteness determination. Appx23367–23368. Though Finjan had only been able to present full testimony from one expert witness pre-mistrial (and ESET had not presented any at all), ESET reasoned that "Finjan's experts cannot agree on the scope of the term ['small']." Appx23367.

The issue proceeded through briefing in September 2020, and was submitted to the district court. Time passed. Then, in late March 2021, the district court entered an eight-page order agreeing with ESET that ***Finjan*** had not established the definiteness of "small." Appx14–21 ("the Indefiniteness Order") (emphasis added.)

The Indefiniteness Order reasoned that "small" was a term of degree that could meet § 112 definiteness standards only if a person of ordinary skill could identify "an objective standard for the size of a 'small' program." Appx19. The Order then analyzed the trial testimony of expert witness Eric Cole, Ph.D., whom Finjan had tendered as part of its infringement case before the mistrial. The Indefiniteness Order noted Dr. Cole's testimony that an ordinary artisan would have understood "small" to mean that a program application "did not require installation," and was "self contained." Appx20 (quoting Appx23164 (414:13–23), Appx23207 (457:8–16)). The district

court, however, found this testimony unpersuasive because, according to the court, Dr. Cole had not presented documentary support for it:

> Dr. Cole testified that a small executable is an application that does not require installation[,]is "self-contained" and is "just running automatically" which is "typical if you go to any website nowadays," whereas an executable that is not small "requires installation" and has "a lot of shared libraries and DLLs and other programs" in order to run. . . . Dr. Cole did not however provide support from the specification, the prosecution history, or from any extrinsic sources in the relevant time period, for this new explanation that a skilled artisan in 1997 would understand "small" to be "uninstalled" or "not requiring installation."

Appx20–21. Based on the above, the district court determined that Finjan had failed to present "evidence of a reasonable range for the size of a small executable or interpretable application program as understood by a skilled artisan in 1997 based on examples provided in the patent specification. Appx21. On that basis, the court declared the term "Downloadable" to be indefinite, and the asserted claims invalid. *Id.*

5. Denial of Reconsideration, Final Judgment, and Appeal

Finjan moved for reconsideration, citing both legal and evidentiary concerns. Appx23476–23477. Legally, Finjan contended that the Indefiniteness Order had misallocated the burden of proof, because the opinion rested on a determination that ***Finjan*** had not presented the required

evidence. Appx23480–23482. As the party challenging validity, ESET—not Finjan—has the burden of proof, and ESET had fallen far short of mustering clear and convincing evidence of indefiniteness. Finjan reminded the court that ESET's invalidity expert, who had not yet testified at the trial, had never opined that the claims were indefinite. *Id.* In fact, he had no trouble understanding and applying the court's construction of "downloadable" when analyzing the prior art. *Id.* Finjan also pointed out that it had had no opportunity to present validity-oriented testimony before the district court declared a mistrial. Appx23482–23485. The Indefiniteness Order's determination that Finjan had "never" offered evidence rebutting ESET's indefiniteness attack was, at best, incomplete. Had the trial proceeded, Finjan would have presented rebuttal testimony, and Finjan attached an offer of proof demonstrating how several witnesses (based on deposition testimony and reports) were ready to give the necessary testimony. Appx23489–23499.

The district court denied Finjan's motion for reconsideration. Appx22–24. It disagreed that the Indefiniteness Order suffered either legal or evidentiary errors. *Id.* The district court entered final judgment for ESET. Appx25. This appeal followed.

SUMMARY OF THE ARGUMENT

This appeal presents serial errors by the district court that led to premature—and incorrect—judgment. As often occurs in patent appeals, an initial error on claim construction led to further errors down the line.

Finjan first asks that the Court correct the district court's construction of the claim limitation "Downloadable." Erroneously relying on ancestor patents not asserted in this case, the district court incorrectly held that a "Downloadable," for every claim of every asserted patent, must be "small." This was notwithstanding that no asserted patent describes any Downloadable as "small," in either its figures, its description or its claims. Indeed, two asserted patents define "Downloadable" without using "small," indicating that there is no size restriction. Though an earlier (unasserted) patent had defined a "Downloadable" as "small," the record makes clear that said patent was an outlier, focusing on applets (i.e., "small" Downloadables) rather than Downloadables generally. Finally, the patents in this family used extensive incorporation by reference, so every asserted patent incorporates (one way or another) conflicting definitions of Downloadable. In that context, the narrower "small" Downloadable of the outlier, applet-focused unasserted patents should not control the fate of other patents. Finjan respectfully asks

that the Court correct the district court's claim construction. Doing so would moot the remaining issues of this appeal.

In the event, however, the Court affirms that the asserted patents are restricted by their outlier ancestor to "small" Downloadables, the district court's second set of errors would require attention. The district court held the asserted claims indefinite, reasoning that Finjan (the patent owner, who should have benefited from the presumption of validity) had failed to establish an objective scope for "small." This was clear legal error as to the burden of proof for indefiniteness. Compounding that error, the district court based indefiniteness on a facially incomplete trial record. Due to a COVID-19 mistrial, no party presented even a single witness on validity before the jury was excused. The district court's determination that Finjan had failed to establish definiteness makes no accommodation for the fact that COVID-19 prevented Finjan from attempting to carry that burden (which was never its burden to carry). Still further, the district court declined to be guided by the evidence of definiteness that was already present in the record. Experts on both sides had interpreted and applied "small," and Finjan had submitted an offer of proof describing what additional testimony it could have presented but for the COVID-19 mistrial. The district court's entry of summary judgment favoring ESET, while failing to even take into account such considerations (let

alone granting Finjan the inferences and accommodations required by the summary judgment standard) was error as a matter of law.

The Court should correct on claim construction. No asserted claim requires "small" Downloadables, and the district court erred in holding otherwise. With that correction, the Court should reverse on indefiniteness, vacate the judgment and remand. If the Court does not correct on claim construction, reversal (or at least vacatur) on indefiniteness would still be necessary, while vacating the judgment and remanding. Finjan respectfully requests such relief.

ARGUMENT

I. STANDARD OF REVIEW

This Court applies regional circuit standards to review of a district court's grant of summary judgment. *Nevro Corp. v. Boston Sci. Corp.*, 955 F.3d 35, 37 (Fed. Cir. 2020). This appeal comes from the Southern District of California, within the Ninth Circuit (28 U.S.C. § 41), so review is de novo. *E.g.*, *Rustico v. Intuitive Surgical, Inc.*, 993 F.3d 1085, 1090–91 (9th Cir. 2021) ("We must determine, viewing the evidence in the light most favorable to the nonmoving party, whether there are any genuine issues of material fact and whether the district court correctly applied the relevant substantive law." (quote marks omitted)).

This Court reviews a district court's indefiniteness determinations de novo except for necessary subsidiary fact findings, which it reviews for clear error. *Nevro*, 995 F.3d at 37.

II. THE DISTRICT COURT ERRED WHEN IT DETERMINED THAT THE CLAIMED DOWNLOADABLE MUST BE "SMALL," AND REVERSAL OF THE INDEFINITENESS ORDER IS WARRANTED

The district court erred when it determined that a "Downloadable," for purposes of these claims, must always be "small." Appx3. No asserted patents actually employ the "small" modifier when describing Downloadables, nor do they embrace the district court's restriction in any other way. Further, while the '520 Unasserted Patent does describe a "small" Downloadable, that statement does not overwhelm other, more express statements in the record indicating that there is no "small" restriction in the asserted patents, each of which differs sharply from the '520 Unasserted Patent. The district court erred when it relied on the '520 Unasserted Patent to construe the asserted patents. The Court should reverse that construction and enter a determination that "Downloadable," for all asserted claims, means:

> "an executable application program which is downloaded from a source computer and run on the destination computer"

See Appx1204–1207. As discussed herein, this is the definition expressly given in two asserted patents (the '844 and '780 Patents), and incorporated by reference into the other three.

The district court's error was to assume "Downloadable" had to have the same meaning not only for each asserted patent, but for every invention within the patent family. No party proposed that family-wide approach to claim construction, and the record does not support it.

A. The '520 and '962 Unasserted Patents Have a Unique Descriptive Focus on "Applets"

This is not a case where all the patent documents in a family share the same description, or even the same definition. Several have their own particular emphasis or focus, and the '520 and '962 Unasserted Patents are examples of this. What sets these unasserted patents apart from the other patent documents (and from all the asserted patents) is the focus on "applets" in the description of the '520 and '962 Unasserted Patents.

Although all asserted patents mention applets, the focus on "applets" in the '520 and '962 Unasserted Patents far exceeds that of any other patent documents. That focus begins in column 1, in which the '520 and '962

Unasserted Patents particularly direct the reader to security problems surrounding small applets:

> [S]ecurity systems are typically not configured to recognize computer viruses which have been attached to or masked as harmless Downloadables (i.e., ***applets***). A Downloadable is a ***small*** executable or interpretable application program which is downloaded from a source computer and run on a destination computer.

Appx1016 (1:29–34 (emphasis added)); Appx1047 (1:35–41 (same)). This description of security risk resembles, but is different from, the description in the other patents. The '520 and '962 Unasserted Patents add the words "applet" and "small" when establishing their invention's background. No other patent at issue does this.

The focus on applets continues in the next paragraph of the '520 and '962 Unasserted Patents. This is a paragraph not present in either the '639 Provisional or any asserted patent. Per this paragraph, the '520 and '962 Unasserted Patents uniquely discuss how computer engineers face "security holes in Downloadable interpreters," using the "Java™ interpreter" as an example. Appx1016 (1:39–51), Appx1047 (1:46–58 (same)). Here, again, the '520 and '962 Unasserted Patents are directing the reader's attention to security problems surrounding applets.

The attention to "applets" continues as the description gets more detailed. Figure 3 of the '520 and '962 Unasserted Patents—a figure not reproduced in any of the asserted patents—depicts a security system that works alongside "a Java™ virtual machine" that can, among other things, "load[ ] applets[.]" Appx1011 (fig.3), Appx1017 (3:42–54); Appx1042 (fig.3), Appx1048 (3:52–63 (same)). Figure 3 describes a security system that whenever a "new applet" requests certain services, the system will "interrupt the request" and pass it on for further analysis. Appx1017 (4:1–8); Appx1048 (4:10–18 (Same)). A "second embodiment" addressing a similar approach to applet security is in figure 4. Appx1012 (fig.4), Appx1018 (5:15–31); Appx1043 (fig4), Appx1049 (5:23–60 (same)). The remainder of the '520 and '962 Unasserted Patents' description addresses flowcharts with additional descriptions of the underlying invention.

None of the later applications—and none of the asserted patents—exhibits such an extensive focus on "applets" and "small" Downloadables as the '520 and '962 Unasserted Patents. No asserted patent repeats, for example, the definitional text highlighting "applets" and characterizing Downloadables as "small." None repeat the recitation of specific security risks for applet interpreters. And none repeat figures 3 and 4, and their applet-focused security techniques.

This is not because the asserted patents ignore "applets." Certainly they are discussed, and the surrounding record makes clear that small applets are a species of "Downloadable" for all asserted patents. But there is no hint in any of the asserted patents that the focus on "applets," or the recitation that Downloadables be "small" could or should carry over into other patents. For those other patents, Downloadables are Downloadables, regardless of whether or not they are "small."

B. Incorporation by Reference Does Not Make "Small" a Requirement of "Downloadables" for All Asserted Patents

The asserted patents' web of incorporations by reference does not restrict "Downloadable" to cover only "small" programs, and the district court erred in reasoning otherwise. "[I]ncorporation by reference does not convert the invention of the incorporated patents into the invention of the host patent." *Modine Mfg. Co. v. U.S. Int'l Trade Comm'n*, 75 F.3d 1545, 1555 (Fed. Cir. 1996). This is because there are always differences between a host patent and the incorporated patent(s), and "in every case, the host patent purports to improve upon that which it incorporates." *Paice LLC v. Ford Motor Co.*, 881 F.3d 894, 908 (Fed. Cir. 2018). Before holding any patent claim beholden to restrictive language in an incorporated patent, it is necessary to examine the full context of the host patent's incorporation.

In this respect, *X2Y Attenuators, LLC v. International Trade Commission*, 757 F.3d 1358 (Fed. Cir. 2014), is instructive. *X2Y*'s issue was whether host patents were bound by disavowals in incorporated patents. It held that context is crucial in determining how an incorporated patent's statements affect claim scope. Per *X2Y*, "it is certainly possible that a clear and unmistakable disavowal in an incorporated patent is no longer so when placed in the context of the disclosure of the host patent." 757 F.3d at 1363. And though this is no disavowal case, *X2Y*'s context-sensitive approach should apply here. *Also cf. Young Dental Mfg. Co. v. Q3 Special Prods., Inc.*, 112 F.3d 1137 , 1143 (Fed. Cir. 1997) ("The specification that is relevant to claim construction is the specification of the patent in which the claims reside."). Additionally, the Court has instructed that the addition/removal of various language from patent specifications as they evolve from a provisional towards issued patents is meaningful for claim analysis. *MPHJ Tech. Invs., LLC v. Ricoh Ams. Corp.*, 847 F.3d 1363, 1369 (Fed. Cir. 2017) ("[I]t is the deletion from the '798 Provisional application that contributes understanding of the intended scope of the final application.").

District courts considering the issue have applied a similarly context-aware analysis. One such case is *Smartphone Technologies LLC v. Research in Motion Corp.*, No. 6:10-cv-74, 2012 WL 489112 (E.D. Tex. Feb. 13, 2012).

There, as here, it was proposed that an asserted patent should be limited by the teachings of an incorporated patent, notwithstanding that the asserted patent had an unambiguous definition of the disputed term ("synchronization"). 2012 WL 489112, at *4. The Eastern District of Texas held that, with such a context, the incorporated patent did not restrict the meaning of "synchronization." *Id.* Another is *Asics America Corp. v. Akeva L.L.C.*, No. 1:09-cv-135, 2015 WL 1013645 (M.D.N.C. Mar. 9, 2015). There, the Middle District of North Carolina noted (citing *X2Y Attenuators*) that the full context of the prosecution history must be taken into account before applying claim restrictions based on incorporated priority applications. 2015 WL 1013645, at *4.

The '520 and '962 Unasserted Patents did state a narrower definition for Downloadable than either the '639 Provisional, the later filed '844 Patent, or the '780 Patent, which rely on a different definition and description. Examining context (as per *X2Y*), the record confirms that, though there was incorporation by reference, the narrow "small" definition of "Downloadable" may suit the '520 and '962 Unasserted Patents, but would not make sense in any asserted patent. The inventors elected not to use the same language in the later filed patents: no asserted patent ever uses the term "small" for Downloadables, or ever suggests a size restriction.

1. The Later Filed '844 and '780 Patents Expressly Restate a Broader Definition of "Downloadable," and Nothing in their Context Suggests Adoption of the "Small" Restriction

Both the '844 and '780 Patents include an express definition of "Downloadable" that omits the "small" restriction. The '844 and '780 Patents make clear that "applets" are included within the scope of "Downloadable," but have no hint that "Downloadable's" scope is limited to "small" programs:

> ***A Downloadable is an executable application program, which is downloaded from a source computer and run on the destination computer.*** . . . Examples of Downloadables include Java™ applets designed for use in the Java™ distributing environment 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.

Appx178 (1:44–59 (emphasis added)); Appx199 (1:50–53 (same)).

Additionally, the '844 and '780 Patents lack the descriptive focus on "applets" observed in the '520 and '962 Unasserted Patents. '844 Figure 5 notably depicts a "generic protection engine 500," which includes—but is not limited to—various security techniques from the '520 Unasserted Patent. Appx174 (fig.5), Appx181 (7:41–8:16 (describing figure 5), 8:17–36 (describing how the engine may "include" components from incorporated

applications)). '780 Figure 7 describes techniques for "decomposing" a Downloadable (no size restriction is given) into an appropriate security profile. Appx197 (fig.7), Appx203 (9:35–57). And none of the figures or description of the '844 and '780 patents are centered on Java "applets," as the description of the '520 and '962 Unasserted Patents were.

With the full context of the host patents taken into account (as required by *X2Y*), it is apparent that the restrictive definition of Downloadable has no place here. The Court should reverse the district court's determination that a "Downloadable," at least for the '844 and '780 Patents, must be "small."

2. The '086 Patent

The Court should apply a similar approach to the '086 Patent. The '086 Patent states no literal definition for "Downloadable" and, as discussed, it has multiple incorporations by reference reaching both of the "Downloadable" definitions at issue here.[7] Yet nothing about the '086 Patent suggests it has an exclusive focus on "small" Downloadables—or that the definition must be

[7] The '086 Patent incorporates by reference the restricted "small" Downloadable definition by incorporating the '962 Unasserted Patent. Appx253 (1:30–35). It incorporates the unrestricted definition by incorporating both the '194 Unasserted Patent and the '780 Patent. *Id.* at 1:20–30.

prescribed to what is stated in the '520 Unasserted Patent. To the contrary, the '086 Patent indicates that it is contemplating Downloadables of every size:

> 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.

Appx253 (2:2–9). By specifically invoking "multiple compressed programs" and other concepts like "application programs" and "Trojan horses," the '086 Patent strongly indicates it does not share the '520 Unasserted Patent's focus on small applets.

The '086 SUMMARY OF THE INVENTION underlines that the '086 Patent is not concerned only with "small" programs, but rather promotes flexibility, and is intended to secure against a wide array of programs, not only applets:

> The present invention provides protection systems and methods capable of protecting a personal computer ("PC") or other . . . 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.***

Appx253 (2:30–43 (emphasis added)). Again, applets are contemplated as "Downloadables" in the '086 Patents, but there are other concepts too, and none invoke a restriction to only "small" programs. In view of such context, incorporating the restrictive "small" interpretation of Downloadable into the '086 Patent would make little sense. The Court should reverse the district court's interpretation of "Downloadable" there, too.

3. The '621 and '755 Patents

As was the case for the '086 Patent, the '621 and '755 Patents include incorporations by reference reaching back to both of the definitions under review. But again, it is clear from context that the "small" restriction would make no sense in the '621 and '755 Patents.

The '621 and '755 Patents each include all the text quoted above from the '086 Patent, demonstrating that all three patents' focus exceeds just "small" programs and reaches a broader scope of "Downloadables." *See* Appx290 (2:33–40), Appx290–291 (2:61–3:7); Appx327 (2:33–40), Appx327–328 (2:61–3:7). Thus for the same reasons addressed *supra* for the '086 Patent, the incorporation of the "small" limitation into the '621 and '755 would be counter to the patents' teachings.

Before the district court, ESET pointed out that certain claims of the '621 and '755 Patents refer to "probes" of various types. *See, e.g.*, Appx300 (21:33–52 ('621 claim 1)); Appx337 (22:30–62 ('755 claim 3)). ESET contended that because the only relevant incorporated patent addressing "probes" was the '962 Unasserted Patent, this meant that the '962 Unasserted Patent's definition of "Downloadable" (limiting to "small" programs) should control. *See* Appx894. Besides ignoring that "Downloadable" and "probe" are entirely separate terms, such argument also ignores *X2Y*'s instruction that context matters. Applying that context, it stands to reason that the '621 and '755 Patents relied on one set of incorporated patents to support the "probes" term, but did not adopt the restrictive interpretation of "Downloadables." As already discussed, the '621 and '755 written description makes clear that the claimed invention for those patents was intended to include, but not be limited by, the "small" applets addressed in the '520 and '962 Unasserted Patents. The Court should reverse the district court's determination that the downloadables claimed by the '621 and '755 Patents must be "small."

C. Upon Correcting the District Court's Claim Interpretation, the Court Should Reverse the Indefiniteness Order, Vacate the Judgment, and Remand

The district court's judgment here rested entirely on putative indefiniteness issues surrounding "small." Appx21. For the reasons stated

above, "small" has no place in the interpretation of "Downloadable" for any asserted patent. Should the Court correct that aspect of the district court's construction, it would eliminate the sole basis for the district court's judgment. Finjan respectfully asks that, in that case, the Court reverse the district court's indefiniteness determination, vacate the judgment for ESET, and remand for trial.

III. SHOULD THE COURT DETERMINE THAT THE CLAIMED DOWNLOADABLE MUST BE "SMALL" THE DISTRICT COURT STILL ERRED WHEN IT GRANTED SUMMARY JUDGMENT TO ESET, AND REVERSAL OF THE INDEFINITENESS ORDER IS WARRANTED

As discussed in the previous section, Finjan respectfully submits that "small" has no place in the proper interpretation of any asserted claim. Should the Court agree, then further argument over the district court's judgment would be moot. Should the Court determine, however, that any asserted claim does require a "small" Downloadable, other errors by the district court confirm that reversal of the Indefiniteness Order would still be warranted. The Order is incompatible with both the Rules of Civil Procedure and the law of indefiniteness. The Court should reverse, vacate the judgment, and remand for further proceedings.

A. The District Court Reversibly Erred When It Placed the Burden of Establishing the Definiteness of "Small" on Finjan Without First Requiring ESET to Come Forward with Evidence of Skilled Artisans' Inability to Understand "Small"

The Indefiniteness Order analyzes the issues as if it were Finjan's burden to establish definiteness, and not ESET's burden to establish the opposite. *See* Appx14–21. When it took the view that Finjan's definiteness submission was lacking, the Indefiniteness Order declared the claims invalid without ever considering whether ESET had mounted a colorable validity attack in the first place. This amounted to the district court applying a presumption of invalidity, rather than the presumption of validity that the Patent Act requires. The Court should reverse.

As the Court knows, a patent is indefinite, and invalid, if "its claims, read in light of the specification delineating the patent, and the prosecution history, fail to inform, with reasonable certainty, those skilled in the art about the scope of the invention." *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 912 n.10 (2014) [*Nautilus I*]. Because patent claims are presumed valid by statute (*see* 35 U.S.C. § 282), one who challenges definiteness bears the burden to prove, with clear and convincing evidence, how and why skilled artisans would run into *Nautilus* problems when interpreting the claim. *Id.* at

912 n.10; *see also One-E-Way, Inc. v. Int'l Trade Comm'n*, 859 F.3d 1059, 1062 (Fed. Cir. 2017).

The Indefiniteness Order cannot be reconciled with that standard. Though citing *Nautilus* (Appx18), the Order does not mention the presumption of validity, nor the challenger's burden of proof. Nor does it discuss any opinion testimony tendered by ESET, either at trial or via declaration.

Instead, the Indefiniteness Order finds various faults with the trial testimony of Eric Cole, Ph.D., whom Finjan had tendered to give testimony about the infringement issues. The order criticizes Finjan and Dr. Cole for failing to prove the interpretation of "small." Appx19 ("Finjan . . . did not provide an explanation as to how [a] skilled artisan would interpret 'small,' or the context in which it would apply."). Simply put, that approach is backwards.

By statute and precedent, Finjan faces no requirement to tender evidence of definiteness (or any other validity aspect) for any claim except if ESET first carries a burden of production by tendering initial evidence of indefiniteness *See generally Tech. Lic'g Corp. v. Videotek, Inc.*, 545 F.3d 1316, 1326–28 (Fed. Cir. 2008) (describing how the burden of production may shift between parties).

There is no hint in the Indefiniteness Order of the district court requiring an initial presentation by ESET before shifting the burden to Finjan. The Indefiniteness Order does not cite a single piece of extrinsic evidence (e.g., expert opinion) presented by ESET. And though the Indefiniteness Order does not mention it, ESET's own expert Eugene Spafford, Ph.D. never tendered an opinion that "small" rendered any claim indefinite. *See generally* Appx23480–23482 (summarizing Dr. Spafford's statements in his expert report); *see also* Appx23507 (in invalidity analysis, Dr. Spafford describing applets as "small"); Appx23510 (Dr. Spafford again describing applets as small for invalidity purposes); Appx23513 (again, for invalidity purposes, describing applets as "small"); Appx23516 (again referring, for invalidity, to "small" applications). As the record establishes, Dr. Spafford had no trouble understanding and applying the concept of a "small" downloadable when analyzing the prior art. And at deposition, Dr. Spafford described in detail his opinions about what was, and was not, a "Downloadable" under the district court's construction. Appx23519–23522.

The district court seems to have adopted ESET's theory that the indefiniteness of Finjan's claims could be established by no more than a failure by Finjan to tender testimony proving otherwise. Appx19. Under that logic, ***all*** patents would be invalid unless and until their owners came forth

with positive evidence of definiteness. The situation would be a presumption of ***invalidity***, directly contradicting the Patent Act. *See* 35 U.S.C. § 282 ("A patent shall be presumed valid.").

The district court's error here is potentially explainable by the unique circumstances of the COVID-19 mistrial, which meant that ESET never presented its invalidity case at trial. But that explanation does not excuse the Indefiniteness Order's failure to properly apply the presumption of validity. The Court should reverse due to plain legal error by the district court.

B. The District Court Reversibly Erred When It Granted Summary Judgment Based on a Facially Incomplete Trial Record

Even were the Court to set aside the errors concerning the presumption of validity, reversal would still be warranted because the record was insufficient as a matter of law to support summary judgment. As the Court knows, summary judgment is only appropriate if there are no material facts in dispute. In other words, ESET should have had to convince the court that no material dispute existed, and the scope of Finjan's claims was so uncertain as to rely on the "unpredictable vagaries of any one person's opinion." *Presidio Components, Inc. v. Am. Tech. Ceramics Corp.*, 875 F.3d 1369, 1377 (Fed. Cir. 2017) (quote marks omitted). Yet at the time of mistrial, there had been no presentation, from either side, addressing the interpretation of "small" from a

validity perspective. On such a backdrop, it was categorically impossible for ESET to carry its heavy summary judgment burden.

At the time the district court declared a mistrial, Finjan was still presenting its case-in-chief. The only patent for which Finjan was able to tender a presentation was the '844 Patent, where Finjan presented infringement testimony from computer scientist Eric Cole, Ph.D. Dr. Cole had not prepared or presented opinions on validity. Finjan had planned to cover that territory later on, with testimony from a different scientist, Michael Goodrich, Ph.D. *See* Appx23482 (describing plan to call Dr. Goodrich). But because of the mistrial, Finjan never got that opportunity. Neither Dr. Goodrich, nor ESET's presumed expert on indefiniteness Eugene Spafford, Ph.D., (*see id.*) ever took the stand.

Regardless of the above, the district court treated the trial record as if both sides had had a full and fair opportunity to present their evidence. The Indefiniteness Order describes ESET's contention that "the trial testimony of Dr. Eric Cole did not remedy" the alleged indefiniteness issue, but ignores that Dr. Cole was not Finjan's only planned witness. Had there been no mistrial, Finjan would have presented evidence from Dr. Goodrich specifically addressing the invalidity issue. Appx23482. The Indefiniteness Order never explains how the district court managed to find an absence of material fact

questions, when the simple fact of the mistrial meant that all the evidence had not yet been heard.

Later on, the district court reasoned that the experts Finjan had planned to present at trial (whom the district court had not actually heard testify) had "no consensus among them as to the scope of the claim term[.]" Appx24. This was based on "the Rule 26 reports and deposition testimony[.]" Appx23. Even assuming *arguendo* an absence of expert consensus prior to trial, Rule 56 does not require a non-movant to establish consensus. It only requires demonstration of a material fact question. Once the district court determined, in its own words, that there was not "consensus" as to the interpretation of "small," that should have been immediate grounds for denying ESET's motion—particularly when one considers the presumption of validity, discussed *supra*. This is yet another basis for reversal.

C. The District Court Reversibly Erred When It Determined That, on the Available Record, ESET had Proved Finjan's Claims Indefinite

Finally, should the Court not reverse on the basis that the district court failed to correctly assign the burden of proof, and should it determine that the incomplete mistrial record was sufficient to support summary judgment, reversal would still be warranted because the district court clearly erred in determining that ESET had presented clear and convincing evidence of a

person of ordinary skill in the art's inability to ascertain the scope of claims requiring a "small" Downloadable.

The "small" restriction is a term of degree. Such terms are definite where the patent provides "some standard for measuring that degree." *Biosig Instruments, Inc. v. Nautilus, Inc.*, 783 F.3d 1374, 1378 (Fed. Cir. 2015) [*Nautilus II*] (quote marks omitted). The standard is not "mathematical precision." *Sonix Tech. Co. v. Publications Int'l Ltd.*, 844 F.3d 1370, 1377 (Fed. Cir. 2017). A term of degree is sufficiently definite where the patent provides "guidance as to the scope of the claims, including, *inter alia*, examples of [covered] structures and criteria for their selection." *Id.* Per *Nautilus I*, "the certainty which the law requires in patents is not greater than is reasonable, having regard to their subject matter." *Nautilus, Inc. v. Biosig Instruments, Inc.*, 572 U.S. 898, 910 (2014). Where, as here, "when a claim language is defined in 'purely functional terms,' a determination of whether the limitation is sufficiently definite is 'highly dependent on the context (e.g., the disclosure in the specification and the knowledge of a person of ordinary skill in the relevant art area).'" *Nautilus II*, 783 F.3d at 1378 (quoting *Halliburton Energy Servs., Inc. v. M-I LLC*, 514 F.3d 1244, 1255 (Fed. Cir. 2008)).

1. The Intrinsic Evidence and Truncated Trial Record Alone Demonstrate Why Summary Judgment Should Have Been Denied

Even limiting review to the truncated mistrial record, the district court clearly erred. The '520 and '962 Unasserted Patents give clear examples of "small" Downloadables such as Java applets and ActiveX controls. Appx1016 (1:24–38); Appx1047 (1:35–45). By design, these are programs that are designed to be interpreted line-by-line when run; they do not require separate installation on a user's system. *See* Appx1017 (3:42–54) (describing how Java-type Downloadable applets are run via a Java virtual machine); Appx1018 (5:15–30 (similar, for ActiveX controls)); Appx1048 (3:52–63 (covering both Java and ActiveX)).

This is consistent with Dr. Cole's testimony during cross-examination before the mistrial. Dr. Cole stated, "the definition of small is an executable that doesn't require installation when it runs on your system[.]" Appx23146 (396:2–5); *see also* Appx23146–23147 (396:19–397:2 ("[A] small [ ] Downloadable, someone skilled in the art is going to deem that as a Downloadable that doesn't require installation. So if you go to a web site it's going to run without installation. . . . [S]omething that doesn't require installation would be a small executable.")). Counsel for ESET challenged Dr.

Cole as to whether there was a specific size cut-off for "small," and Dr. Cole explained why there was not. Appx23147 (397:14–18).

This evidence alone—the teachings of the '520 and '962 Unasserted Patents, interpreted by Dr. Cole during cross-examination, were more than enough to establish a material fact dispute about whether skilled artisans would have been able to understand the scope of "small." The district court clearly erred in determining otherwise. Even limiting consideration to the teachings of the '520 and '962 Unasserted Patents and the cross-examination testimony of Dr. Cole, Finjan showed the district court examples of "small" Downloadables and the criteria for their selection. No more is required to survive summary judgment. *Sonix Tech.*, 844 F.3d at 1377 (describing how "examples of [covered] structures and criteria for their selection was sufficient to establish definiteness" (citing *Enzo Biochem, Inc. v. Applera Corp.*, 599 F.3d 1325, 1336 (Fed. Cir. 2010))).

2. Opinion Testimony That Finjan Never Had a Chance to Present at Trial Further Confirms Definiteness

The Indefiniteness Order makes no consideration of the testimony that would have been presented but for the mistrial. Consideration of such testimony, which Finjan presented when seeking reconsideration of the

Indefiniteness Order, further demonstrates the district court's error. *See generally* Appx23480–23485.

Testimony that ESET's own expert was able to interpret "small." In his expert report and at deposition, ESET's expert Eugene Spafford, Ph.D. never opined that a skilled artisan would be unable to interpret "small," and indeed gave numerous examples of "small" Downloadables. *See generally* Appx23481 (recounting Dr. Spafford's testimony); *see also* Appx23507 (in invalidity analysis, Dr. Spafford describing applets as "small"); Appx23510 (Dr. Spafford again describing applets as small for invalidity purposes); Appx23513 (again, for invalidity purposes, describing applets as "small"); Appx23516 (again referring, for invalidity, to "small" applications). At trial, Finjan could have examined Dr. Stafford on these opinions, which fully corroborate how the examples in the asserted patents illuminate the scope of "small." Still further, Dr. Spafford conceded at deposition that "small" Downloadables could include "Java outputs," "ActiveX," and "JavaScript." Appx23495–23497; *see also* Appx23518–23522. Again, but for the COVID mistrial, Finjan could have presented those concessions to the jury via cross-examination.

Testimony from other expert witnesses Finjan had planned to present. Finjan also submitted to the district court an offer of proof

indicating additional testimony from three other experts on the interpretation of "small." Appx23489–23494. Again, Finjan was ready to present these experts at trial, but was cut off by the COVID-19 mistrial. Appx23489. The district court rejected the offer as demonstrating "the lack of consensus among its own experts in construing ['small']," and so declined to be guided by it. Appx24. This was clear error as well.

The three experts Finjan was barred from presenting were Dr. Goodrich (mentioned *supra*); computer scientist Nenad Medvidovic, Ph.D.; and computer scientist Trent Jaeger, Ph.D. Appx23489–23494. Each of them described how the meaning of "small" was ***context-dependent*** and generally described a Downloadable that was small in comparison to other programs such as a browser:

- Dr. Goodrich would have described how a person of ordinary skill would appreciate, from the '520 and '962 Unasserted Patents, that the status of being "small" was relative and context dependent. He described how the '962 Unasserted Patent, in particular, "gives several examples of this," including the Java applets and ActiveX controls discussed above. Appx23490–23493; *see also* Appx23525–23532.

- Dr. Medvidovic would have described, similarly, how (to a person of ordinary skill) common programs like Microsoft Word, PowerPoint, and Internet Explorer would be too large to fall within "small," as defined by the patents. Appx23493–23494; *see also* Appx23534–23535.
- Dr. Jaeger would have similarly described how large files like a mail database or mail files would be too large to be within "small." Appx23494; *see also* Appx23537–23540.

The district court declined to be guided by any of this evidence, rejecting it as exhibiting a "lack of consensus." Appx24. Finjan does not agree that there was any lack of consensus—all the testimony above treats "small" as a reasonably understandable term, buttressed by real-world examples. Regardless, on a motion for summary judgment where Finjan's only burden was to identify a material fact question, and not to establish certainty, these considerations should have been more than enough to warrant denial. The district court clearly erred in holding otherwise, and the Court should reverse.

D. The Court Should Reverse the Indefiniteness Order, Vacate the Judgment, and Remand

The myriad issues in the Indefiniteness Order—its failure to apply the correct standard of proof, its reliance on a facially incomplete evidentiary

record, and its clear error in fact-finding—compound. Any one of them alone would be grounds to reverse, and taken together the case for reversal is overwhelming. The Court should reverse, vacate the judgment, and remand.

CONCLUSION

For the reasons stated herein, Finjan respectfully asks that the Court correct the district court's claim construction, reverse the Indefiniteness Order, vacate the judgment for ESET, and remand for further proceedings.

Dated: August 23, 2021

Respectfully submitted,

/s/ Juanita R. Brooks

Juanita R. Brooks
FISH & RICHARDSON P.C.
12860 El Camino Real, Suite 400
San Diego, CA 92130
Telephone: (858) 678-5070

Robert Courtney
FISH & RICHARDSON P.C.
60 South 6th Street, Suite 3200
Minneapolis, MN 55402
Telephone: (612) 335-5070

ADDENDUM

UNITED STATES DISTRICT COURT

SOUTHERN DISTRICT OF CALIFORNIA

FINJAN, INC., Plaintiff, v. ESET, LLC, a California Limited Liability and ESET SPOL. S.R.O., a Slovak Republic Corporation, Defendants.	Case No.: 3:17-cv-0183-CAB-(BGS) **CLAIM CONSTRUCTION ORDER**

On September 25 and 26, 2017, the Court held a hearing to construe certain terms and phrases of the following patents: U.S. Patent Nos. 6,154,844; 6,804,780; 8,079,086; 9,189,621; 9,219,755; and 7,975,305. The parties submitted briefing in accordance with this District's local patent rules and the case management order. A tutorial was presented by both sides to assist the Court with the history and background of the patents.

The Court requested further briefing regarding certain terms. Having now considered all the submissions of the parties, the arguments of counsel and for the reasons set forth at the hearing and herein, the Court hereby enters the claim constructions set forth below.

A. "Downloadable" in U.S. Patent Nos. 6,154,844; 6,804,780; 8,079,086; 9,189,621 and 9,219,755

The parties seek construction of the term **Downloadable** as it is used in five of the patents at issue. This claim term can be traced through two branches of the family tree of this patent (*see* Attachment A) with somewhat differing definitions. The Court however concludes that the variations between the definitions can be reconciled and the specifications in their entirety give notice of what is encompassed by the claim term **Downloadable** to one of skill in the art.

Downloadable initially appears as a defined term in the specification of the 6,167,520 patent, and its continuation the 6,480,962 patent, as *a small executable or interpretable application program which is downloaded from a source computer and run on a destination computer.*

The specification of the 6,092,194 patent, and its continuation the '780 patent (which is at issue in this litigation), define **Downloadable** *as an executable application program which is downloadable from a source computer and run on the destination computer.* The specification however provides as examples of a **Downloadable**, application programs such as Java™ applets, known as little application programs in machine language, and JavaScripts™ scripts, an interpretable application program. These examples are in accord with the definition, incorporated by reference, set forth in the '520 patent. The '194 patent and its progeny therefore conform to the '520 patent's definition of **Downloadable** as *small executable or interpretable application programs* through the use of the examples in the specifications. The Court finds that one of skill in the art would be able to ascertain what is claimed as a **Downloadable** in the context of these patents, and that in light of the examples provided in the '194 patent specification, concludes that the meaning of **Downloadable** is consistent with the definition provided in the '520 patent.

The '844 patent (which is at issue in this litigation) defines **Downloadable** *as an executable application program which is downloadable from a source computer and run on the destination computer* and also includes references to small executable and

interpretable application programs as examples of a **Downloadable**. The '844 patent incorporates by reference the '520 patent and '194 patent. The Court finds that the definition of **Downloadable** based on the '844 patent specification, the examples provided therein and the incorporation of the '520 patent and the '194 patent, informs one of skill in the art with reasonable certainty the scope of the invention. The entirety of the specification's description would inform that **Downloadable** includes executable and interpretable application programs, in accordance with the examples and incorporated references.

The '822 patent is a Continuation in Part of both the '962 patent and '780 patent and incorporates those patents by reference. Its continuation patents, including the '086 patent, '621 patent and '755 patent, which are at issue in this litigation, do not include a definition of **Downloadable** in the specification but incorporate by reference the '962 patent and the '780 patent, and their definitions and examples of a **Downloadable**.

The Court finds that the two branches of the family tree of the patents at issue inform that a **Downloadable** in the context of these patents means a *small executable or interpretable application program which is downloaded from a source computer and run on a destination computer*. This construction comports with the plain definition set forth in the '520 patent and the '962 patent, and is supported by the written description including the definition and the examples set forth in the '194 patent and its progeny, and in the entirety of specification of the '844 patent.

The Court therefore construes the term **Downloadable** in all five patents as ***a small executable or interpretable application program which is downloaded from a source computer and run on a destination computer.***

B. U.S. Patent No. 6,154,844

The parties sought construction of the following phrase appearing in various claims of the '844 patent: **before the web server makes the Downloadable available to web clients**. The Court's construction for this term is: ***Before the Downloadable is available on a web server to be called up or forwarded to a web client.*** ('844 @ Col. 3:32-52; Col. 4:65 - Col. 5:13; Figure 1.)

C. U.S. Patent No. 6,804,780

The parties sought construction of the following terms and phrases appearing in various claims of the '780 patent. The Court's constructions for these terms are:

Claim Term	COURT'S CONSTRUCTION
software components required to be executed by the Downloadable	***components of code that the Downloadable is required to execute*** (agreed construction)
ID generator	Defendant's request for application of 112 ¶6 denied. "ID generator" is not a nonce term as advocated by Defendant. It is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier.
performing a hashing function on the Downloadable and the fetched software components to generate a Downloadable ID	***performing a hashing function on the Downloadable together with its fetched software components*** (Adopting PTO Construction from the IPR of the '780 patent April, 2016.)

D. U.S. Patent No. 7,975,305

The parties sought construction of the following phrase appearing in various claims of the '305 patent, **network interface, housed within a computer.** Defendant's proposed construction that "housed within a computer" necessarily limits the structure of the network interface to a hardware component is declined. The specification includes software interface examples. The Court therefore finds in the context of the patent, the use of "housed" in contrast to "stored" does not dictate that the claim be limited to hardware components. To the extent clarification is required the Court construes this phrase as ***network interface, contained within the computer.***

The parties agreed construction for **database**, ***a collection of interrelated data organized according to a database schema to serve one or more applications***, is adopted.

E. U.S. Patent No. 8,079,086

The parties sought construction of the following terms appearing in various claims of the '086 patent. The Court's constructions for these terms are:

Claim Term	COURT'S CONSTRUCTION
appended Downloadable	***a Downloadable with a representation of the DSP data attached to the end of the Downloadable*** Declaration of Dr. Spafford, ¶¶36-39, and references cited therein, that one skilled in the art at the time would understand "append" to mean attach or add to the end of the existing file. The claim recites appending a representations of the DSP data to the Downloadable indicating an order.
destination computer	***Separate computer receiving the appended Downloadable***

file appender	Defendant's request for application of 112 ¶ 6 is denied. "File appender" is not a nonce term as advocated by Defendant. It is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier.
Transmitter	Defendant's request for application of 112 ¶ 6 is denied. "Transmitter" is not a nonce term as advocated by Defendant. It a common name for a known program construct that would be familiar to one of skill in the art.

F. U.S. Patent No. 9,189,621

The parties sought construction of the following terms appearing in various claims of the '621 patent. The Court's constructions for these terms are:

Claim Term	COURT'S CONSTRUCTION
wherein the information pertaining to the downloadable includes information pertaining to operation of the downloadable and distinct from information pertaining to the request	Defendant asserts that this phrase in indefinite as it is not possible to ascertain the meaning of "information pertaining to the operation of the downloadable that is distinct from information pertaining to the request." The Court finds that in the context of the claim in its entirety and for the reasons set forth on the record, one of skill in the art would understand the meaning and scope of this claim language, and no further construction is needed.
a response engine for performing a predetermined	Defendant asserts that use of "response engine" is the equivalent of "means for" claiming and is

responsive action based on the comparison	limited to the structures disclosed in the '520 Patent at Figs. 5 and 6, Steps 525, 530, 540 and 610-30. Plaintiff asserts that "engine" is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier, in this case "response." The Court agrees that "engine" is not a nonce term as advocated by Defendant, and that the claim provides sufficient structure for one skilled in the art. Defendant's request for application of 112 ¶ 6 is denied.
a response engine for performing a predetermined responsive action based on the comparison with the information pertaining to the predetermined suspicious downloadable	Defendant asserts that "response engine" is the equivalent of "means for" claiming and no structure to perform this function is disclosed. Plaintiff asserts that "engine" is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier, in this case "response." "Engine" is not a nonce term. Defendant's request for application of 112 ¶ 6 is denied.
Database	***a collection of interrelated data organized according to a database schema to serve one or more applications*** (joint construction)

The final construction at issue with regard to the '621 patent is a "means for" limitation. An element of a claim may be expressed as a means for performing a specified function without the recital of the structure and is construed to cover the corresponding structure described in the specification or equivalents thereof. 35 U.S.C. § 112, ¶ 6. The limitation at issue appears in claim 15, which is dependent on claim 10.

> Claim 10. A system for reviewing an operating system call issued by a downloadable, comprising:
> …
> a plurality of operating system probes for monitoring substantially in parallel a plurality of subsystems of an operating system during runtime for an event caused from a request made by a Downloadable, wherein the plurality of subsystems includes a network system;
> ….
> Claim 15. The system of claim 10, wherein the plurality of operating system probes operating substantially in parallel for monitoring the operating system includes **means for monitoring a request sent to a downloadable engine.**

The parties agree that the "means for" element of claim 15 is governed by 35 U.S.C. § 112, ¶ 6. The patent specification must disclose with sufficient particularity the corresponding structure for performing the claimed function and clearly link that structure to the function. *Triton Tech of Tx., LLC. V. Nintendo of Am., Inc*., 753 F.3d 1375, 1378 (Fed. Cir. 2014). The claimed function is "monitoring a request sent to a downloadable engine." After the claim construction hearing, the Court requested supplemental briefing from the parties identifying with sufficient particularity the corresponding structure in the specification for performing the claimed function. [Doc. 178-1.]

The plaintiff refers to the specification of the '962 patent [Doc. No. 138-9], incorporated by reference in the '621 patent, for the corresponding structure that supports this "means for" claim. Plaintiff directs the Court to the component identified in the specification as the request broker 306 described at Col. 4:12-18, Figs. 3 and 4 [*id.* at 6-7, 12] as the corresponding structure providing the function in the system of monitoring a request sent to a downloadable engine. [*See* Doc. No. 183 at 3.]

The specification describes the system for monitoring requests made by a Downloadable 140 to a downloadable engine 250 at Col. 3:51 – Col. 5:48. [Doc. No. 138-9, at 6-7, 12-13.] In the examples provided (*see* Figs. 3 and 4), extensions 304, 404, 405, 406 examine a Downloadable's request for access to classes 302 of a Java™ Virtual Machine (the downloadable engine 250) or to message calls 401, DDE calls 402 and DLL calls 403 of a ActiveX™ platform (the downloadable engine 250). The Downloadable's request to the downloadable engine may be interrupted by the extension which then notifies the request broker 306 of the Downloadable's request. The request broker 306 in turn forwards the request to the event router 308. [*Id.*]

The extensions 304, 404, 405, 406 monitor requests made to the downloadable engine 250. Col. 5:40-49 (the extensions recognize a request made by a Downloadable to the downloadable engine, interrupt the processing of the request and generate and forward a message identifying the incoming Downloadable to request broker which forwards the message to the event router.); Col. 4:10-15 (each extension 304 manages a respective one of the Java™ classes, interrupts the request and generates a message to the request broker 306); Col. 5:23-38 (each extension 404, 405, 406 recognizes a call to an ActiveX™ platform 401, 402, 403, and generates a message to the request broker 306). The request broker 306 forwards the request on for further analysis and response. The structures identified in the specification corresponding to a **means for monitoring a request sent to a downloadable engine**, are the Java Class extensions 304, the Message Extension 404, the Dynamic-Data-Exchange Extension 405 and Dynamically-Linked-Library Extension 406, and their equivalents.

F. U.S. Patent No. 9,219,755

The parties sought construction of the following terms appearing in various claims of the '755 patent. The Court's constructions for these terms are:

Claim Term	COURT'S CONSTRUCTION
a downloadable engine for intercepting a request message being issued by a downloadable to an operating system	Defendant asserts that "downloadable engine" is the equivalent of "means for" claiming and the disclosed structures disclosed to perform this function are a Java Virtual Machine 250 or Active X Platform 250 ('960 patent, Col. 3:54-56; Fig. 3; Col. 5:25-27; Fig. 4). Plaintiff asserts that "engine" is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier, in this case "response." The Court agrees that "engine" is not a nonce term as advocated by Defendant, and that the claim provides sufficient structure for one skilled in the art. Defendant's request for application of 112 ¶ 6 is denied.
intercepting an operating system call being issued by the downloadable to an operating system	***stopping a request message before the request message is received by the operating system***
a response engine for receiving a violation message from the runtime	Defendant asserts that use of "response engine" is the equivalent of "means for" claiming and is limited to the structures disclosed in the '520

environment … and blocking … and allowing ….	Patent, at Figs. 5 and 6, Steps 525, 530, 540 and 610-30. Plaintiff asserts that "engine" is a common name for a known program construct that would be familiar to one of skill in the art to perform a function further identified by its modifier, in this case "response." The Court agrees that "engine" is not a nonce term as advocated by Defendant, and that the claim provides sufficient structure for one skilled in the art. Defendant's request for application of 112 ¶6 is denied.

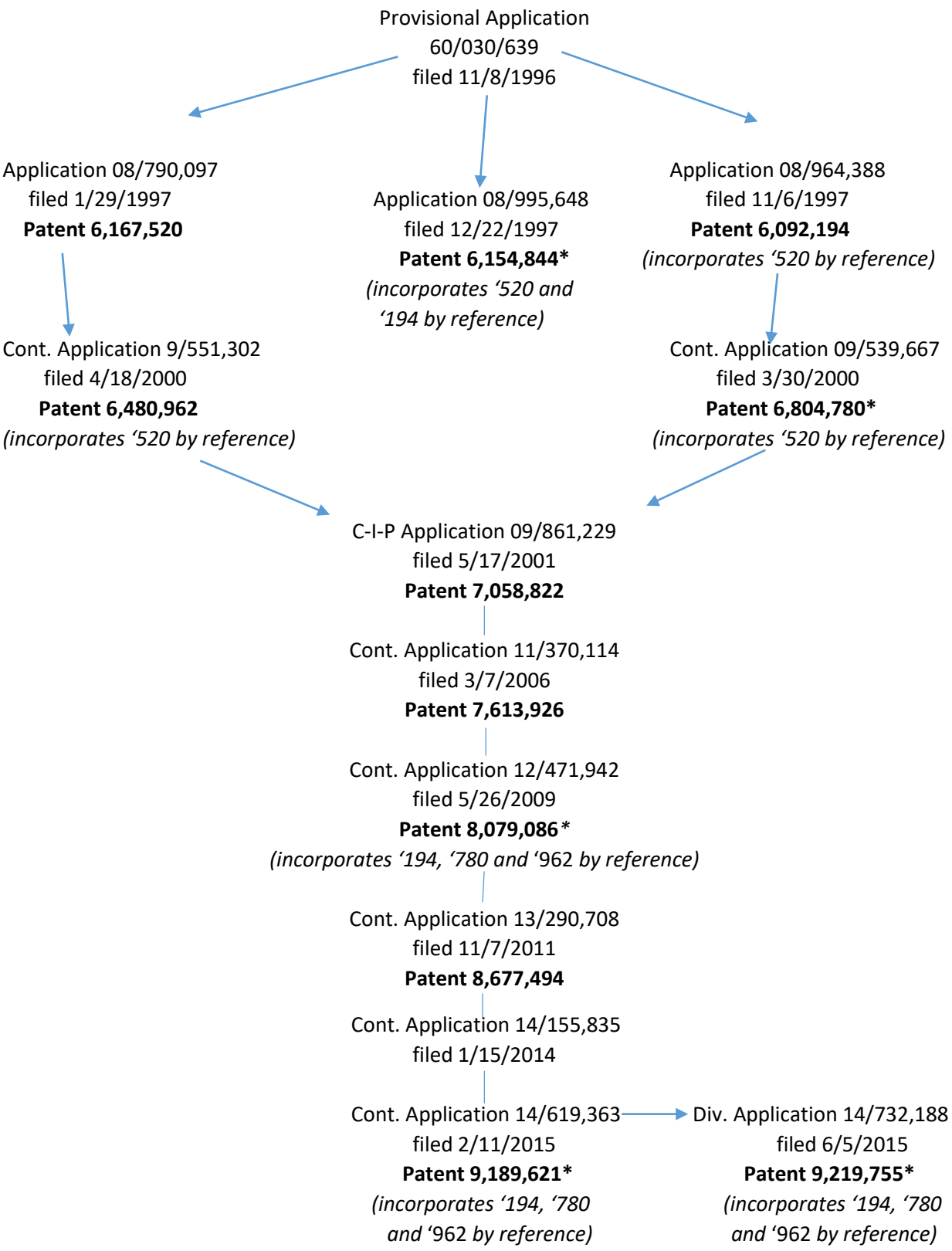
It is **SO ORDERED**.

Dated: November 14, 2017

Hon. Cathy Ann Bencivengo
United States District Judge

Attachment A

ATTACHMENT A



*Patents at issue in the litigation.

UNITED STATES DISTRICT COURT

SOUTHERN DISTRICT OF CALIFORNIA

FINJAN, INC.,

Plaintiff,

v.

ESET, LLC, a California Limited Liability and ESET SPOL. S.R.O., a Slovak Republic Corporation,

Defendants.

Case No.: 3:17-cv-0183-CAB-BGS

AMENDED ORDER ON DEFENDANTS' RENEWED MOTION FOR SUMMARY JUDGMENT OF INDEFINITENESS
[Doc. No. 806]

Before the Court is the renewed motion of Defendants ESET, LLC and ESET spol. s.r.o (collectively "ESET") for summary judgment to invalidate Plaintiff Finjan's United States Patent Nos. 6,154,844; 6,804,780; 8,079,086; 9,189,621; and 9,219,755 ("the patents at-issue") as indefinite pursuant to 35 U.S.C. § 112 based on this Court's construction of the claim term "Downloadable." The motion is fully briefed, and the Court deems it suitable for submission without oral argument.

I. Background

This motion has an unusual history. ESET filed a motion for summary judgment asserting that the patents at-issue are indefinite at the close of fact discovery in this case, and the Court held argument on September 26, 2019. Finding that there were factual disputes regarding what a skilled artisan in 1997 would have understood constituted a

"Downloadable" based on the Court's construction of that term, the Court denied the motion without prejudice. [Doc. No. 699.] It was anticipated that trial testimony would establish what was generally understood in the art in 1997 as a "Downloadable" and such testimony would inform the scope of infringement. [Doc. No. 697, at 22:3-15.]

A jury trial commenced in this case on March 10, 2020. After three trial days the Court was forced to vacate the remainder of the trial, excuse the jury and declare a mistrial due to the COVID-19 pandemic and the issuance of the State of California's stay-home order. [Doc. No. 783.] This District's continuing moratorium on civil jury trials and backlog of criminal jury trials currently precludes scheduling a new trial in this matter.

Having heard testimony from Finjan's expert during the vacated trial on this issue, however, the Court permitted ESET to renew this motion in consideration of the testimony that was taken. Although Finjan's patents have been the subject of much litigation, and the term "Downloadable" has been construed by other courts, the issue raised in ESET's current motion does not appear to have been addressed by any prior constructions.

Finjan is the owner of a large family tree of patents for security systems and methods of detecting malware in computer programs. Finjan has litigated many of their patents, including some of the patents at-issue in this motion, in other district courts. Many have also been subject to *inter partes* review by the Patent and Trademark Office (PTO). The Federal Circuit has issued at least nine opinions, precedential and non-precedential, on appeals from district courts and the PTO regarding Finjan patents. Yet none of these orders or opinions discuss how earlier references incorporated into the patents at-issue inform the construction of the term "Downloadable."

II. The Construction of "Downloadable"

In 2017, Finjan filed this litigation against defendants ESET asserting infringement of the patents at-issue, and United States Patent No. 7,975,305.[1] Finjan claims priority for

[1] The '305 patent is not subject to this motion as it does not include the claim term "Downloadable."

the patents at-issue back to an application filed on November 8, 1996, Provisional Application 60/030639. [Doc. No. 139-24.] The application is directed at "a system and method for protecting computers from hostile Downloadables," described as executable application programs automatically downloaded from a source computer and run on the destination computer that might carry computer "viruses." [Id., at 5-6.] The claim term "Downloadable" is presented as a capitalized term in the provisional application and all the patents at-issue, signaling it is a specifically defined term. The definition of "Downloadables," however, is not consistent throughout Finjan's subsequently issued patents. The explicit definitions include:

- "applets" (little applications) described in the 1990s as small interpreted or executable programs. *See* Provisional Application 60/030639 (filed November 8, 1996) [*Id.* at 5-6.]
- "Downloadables (i.e., applets)" as "a small executable or interpretable application program which is downloaded from a source computer and run on a destination computer," in conformity with the original provisional application. *See* U.S. Patent No. 6,167,520, at Col. 1:31-34 (application filed January 29, 1997); U.S. Patent No. 6,480,962, at Col. 1:38-41 (filed April 18, 2000).
- "an executable application program which is downloaded from a source computer and run on a destination computer" (without "i.e., applet," "small" or "interpretable" included in the definition but using applets and interpretable programs as examples of a "Downloadable" and incorporating the earlier definition by reference). *See* U.S. Patent No. 6,092,194, at Col. 1:44-55 (filed November 6, 1997); U.S. Patent No. 6,804,780 at Col. 1:50-60 (filed March 30, 2000).

Other district courts have determined that "Downloadable" lacked ordinary meaning when the patents were filed and construed it as "an executable application program which is downloaded from a source computer and run on a destination computer," applying the

explicit definition from the '194 patent. [Doc. No. 139-10, at 3; Doc. No. 138-4, at 2-5 (the term was not amenable to plain and ordinary meaning and the patent applicant intended to act as the lexicographer of this term, therefore the specification definition controls).] None of these orders, however, discussed the significance of the '520 patent's definition incorporated into the '194 patent and its continuations. One district court, without explanation, applied the broader definition from the '194 patent specification to the construction of the term "Downloadable" in the '962 patent as "the same" definition [id., at 3, fn. 4], disregarding the fact the '962 patent explicitly defines "Downloadable" as "a *small* executable or interpretable application program which is downloaded from a source computer and run on a destination computer." *See* U.S. Patent No. 6,480,962, at Col. 1:39-41 (*emphasis* added).

Incorporation by reference provides a method for integrating material from various documents into a host document by citing such material in a manner that makes clear that the material is effectively part of the host document as if it were explicitly contained therein. *See Trustees of Columbia Univ. v. Symantec,* 811 F.3d 1359, 1365-66 (Fed. Cir. 2016) (*citing Advanced Display Sys. v. Kent State Univ.*, 212 F.3d 1272, 1282 (Fed. Cir. 2000) (provisional applications incorporated by reference are effectively part of the specification as though it was explicitly contained therein.)). By incorporating the earlier definition of "Downloadable" from the '520 Patent into the '194 Patent and subsequent continuations (including the patents at-issue), the scope of the term is limited to "*small* executable or interpretable application programs," and not *all* executable application programs (*emphasis* added). *See Symantec*, 811 F.3d at 1365 (rejecting a broad interpretation of a claim term in part because a provisional application incorporated by reference the same term more narrowly defined.) Inconsistent language used later cannot support a broad claim construction when the explicit definition is incorporated from earlier patents in the family tree.

In this case, the Court concluded that based on its incorporation by reference in all the patents at-issue, the explicit definition of "Downloadables" from the '520 patent and

the '962 patent, which is supported by the examples provided in the specification, is the proper construction of "Downloadables" – "a small executable or interpretable application program which is downloaded from a source computer and run on a destination computer." [Doc. No. 195.]

III. The Indefiniteness Determination

The Court's claim construction, not unexpectedly, resulted in the present dispute as to the scope of the modifier "small." ESET argues that "small" is a term of degree with not technical meaning or defined boundaries and there is insufficient information in the intrinsic record for a skilled artisan to have clear notice of what constitutes a "small executable or interpretable application program." They further argue that this is demonstrated by the inability of Finjan's experts to come to a consistent opinion as to what objective boundaries constitute a small application program. [806-1, at 5, 17.]

The definiteness requirement of paragraph 2 of 35 U.S.C. §112 requires that the "specification shall conclude with one or more claims particularly pointing out and distinctly claiming the subject matter which the applicant regards as his invention." The definiteness requirement focuses on whether "a patent's claims, viewed in light of the specification and prosecution history inform those skilled in the art about the scope of the invention with reasonable certainty." *Nautilus, Inc. v. Biosig Instruments, Inc.,* 572 U.S. 898, 910 (2014). The inquiry "trains on the understanding of a skilled artisan at the time of the patent application." *Id.* at 911.

Terms of degree must provide sufficient certainty to one of skill in the art to afford clear notice of what is claimed and what is still open to the public. *See Biosig Instruments, Inc. v. Nautilus, Inc.*, 738 F.3d 1374, 1378 (Fed. Cir. 2015) ("When a 'word of degree' is used, the court must determine whether the patent provides 'some standard for measuring that degree.'"); *Interval Licensing LLC v. AOL, Inc.,* 766 F.3d 1364, 1370 (Fed. Cir. 2014) (the definiteness standard must allow for a modicum of uncertainty but must also require clear notice of what is claimed thereby apprising the public of what is still open to them).

Finjan's contention that the claim term "Downloadable" cannot be indefinite simply because the Court was able to construe it by adopting the explicit definition provided by the inventor in the '520 patent fails to address the issue raised by ESET. A defined term is still indefinite if a person of ordinary skill in the art cannot translate the definition into meaningfully precise claim scope. *Halliburton Energy Servs., Inc. v. M-I LLC*, 514 F.3d 1244, 1251 (Fed. Cir. 2008). The issue therefore is whether a skilled artisan in 1997 would have understood with reasonable certainty based on the specification and prosecution history what the inventor meant by a "small" application program and therefore understood what comes within the scope of the claims.

The '510 patent, incorporated into all the later patents, describes a Downloadable as an "applet," a small interpretable or executable application program, and provides that "a Downloadable is used in a distributed environment such as the Java™ distributed environment produced by Sun Microsystems or in the Active X™ distributed environment produced by Microsoft Corporation." *See* '520 Patent, at Col. 1:31-32, 34-38. Because such examples existed in the 1990s, there should be an objective standard for the size of a "small" program. While "absolute or mathematical precision" was not required, some objective boundary should be identifiable from the disclosed embodiments. *See Biosig Instruments*, 738 F.3d at 1381.

Finjan opposed ESET's initial motion for summary judgment for indefiniteness as to the scope of "small" on the grounds that a numerical limitation or cut-off is not necessary because a skilled artisan could determine if an application is "small" from the examples in the '962 patent and based on the context. Finjan, however, did not provide an explanation as to how that skilled artisan would therefore interpret "small" or the context that would apply. [Doc. No. 610, at 8 and 15.][2]

[2] Finjan has also argued that the Court should reconsider its incorporation of "small" into the construction of Downloadable and disregard that modifier as other district courts have done to avoid this definiteness issue. [Doc. No. 812, at 6.] This solution may resolve Finjan's problem with defining "small," but the fact that the Court's construction results in indefiniteness is not a basis for reconsideration.

ESET argued then, as it does now, that Finjan's experts did not provide any objective boundaries for a "small" application program based on what a skilled artisan would have understood was upper end of "small" in the context of application programs being downloaded from a source computer to run on a destination computer at the time the application was filed. [Doc. No. 816, at 5.] Finjan's contention that the understanding of what is "small" depends on the context is not supported by the intrinsic evidence or even extrinsic evidence of the state of the art at the relevant time. It amounted to "unpredictable vagaries of any one person's opinion" and therefore failed to provide sufficient notice as to the scope of the term. *Interval Licensing*, 766 F.3d at 1371 ("[A] term of degree fails to provide sufficient notice of its scope if it depends on the unpredictable vagaries of any one persons' opinion.")

ESET contends that the trial testimony of Finjan expert Dr. Eric Cole did not remedy this defect.[3] Dr. Cole presented an explanation how a skilled artisan would interpret "small" that was neither disclosed in his previous declaration to the Court (small meant "a few megs … something that is not multiple gigs or really large" [Doc. No. 806-1, at 17]) or anchored to the specification or prosecution history. Rather than providing a range of application size that would have been construed as "small" by an artisan in 1997, Dr. Cole testified on Finjan's behalf that an application would be understood to be small if it "did not require installation" and opined that "small" depends not on size but on the function. Dr. Cole testified that a small executable is an application that does not require installation is "self-contained" and is "just running automatically" which is "typical if you go to any website nowadays," whereas an executable that is not small "requires installation" and has "a lot of shared libraries and dlls and other programs" in order to run. [Doc. No. 812, at 12.] Dr. Cole testified that regardless of time period, Internet speed and other factors related

[3] Although the trial was terminated early due to the pandemic, Dr. Cole's testimony was completed. Finjan suggests that Dr. Cole's testimony at a future trial will replace his completed testimony, but a subsequent trial is not an opportunity for Dr. Cole to change his opinions or supplement them with support he did not provide on the record at the first trial. [Doc. No. 816, at 11.]

to capacity, what fits the criteria of "small" may change but this distinction (installed or not installed) is constant. Dr. Cole did not however provide support from the specification, the prosecution history, or from any extrinsic sources in the relevant time period, for this new explanation that a skilled artisan in 1997 would understand "small" to be "uninstalled" or "not requiring installation."

In sum, Finjan never offered evidence of a reasonable range for the size of a small executable or interpretable application program as understood by a skilled artisan in 1997 based on examples provided in the patent specification. Instead, Finjan elected at trial to offer a new understanding without reference to the size of the application as the objective boundary of a "small" application. Finjan's new definition is not supported by the specification or prosecution history. It may be convenient to support Finjan's infringement contentions against ESET's accused devices, but Finjan's new explanation does not provide clear notice of what constitutes a "small executable or interpretable application program."

IV. Conclusion

For the foregoing reasons, the Court finds that the term "Downloadable" as used in the patents at-issue is indefinite. Accordingly, it is hereby **ORDERED** that ESET's motion for summary judgment of invalidity of Finjan's United States Patent Nos. 6,154,844; 6,804,780; 8,079,086; 9,189,621; and 9,219,755 for indefiniteness is **GRANTED**.

It is **SO ORDERED**.

Dated: March 29, 2021

Hon. Cathy Ann Bencivengo
United States District Judge

UNITED STATES DISTRICT COURT

SOUTHERN DISTRICT OF CALIFORNIA

FINJAN, INC., Plaintiff, v. ESET, LLC, a California Limited Liability and ESET SPOL. S.R.O., a Slovak Republic Corporation, Defendants.	Case No.: 3:17-cv-0183-CAB-BGS **JUDGMENT ON '844, '780, '086, '621, AND '755 PATENTS**

It is hereby **ORDERED** that **JUDGMENT** in favor of Defendants and against Plaintiff is entered on Plaintiff's claims related to United States Patent Nos. 6,154,844; 6,804,780; 8,079,086; 9,189,621; and 9,219,755. Plaintiff's claims related to United States Patent No. 7,975,305 remain severed and stayed.

It is **SO ORDERED**.

Dated: May 20, 2021

Hon. Cathy Ann Bencivengo
United States District Judge

Exhibit A

US006154844A

United States Patent [19]

Touboul et al.

[11] Patent Number: 6,154,844

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

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

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

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

[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.**[7] **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

[56] **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.

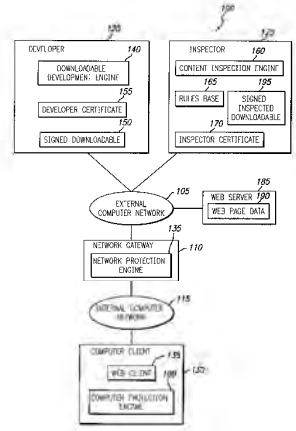
(List continued on next page.)

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

[57] **ABSTRACT**

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.

44 Claims, 7 Drawing Sheets



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/iel$_{13}$cgi?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts, Okamato, 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.

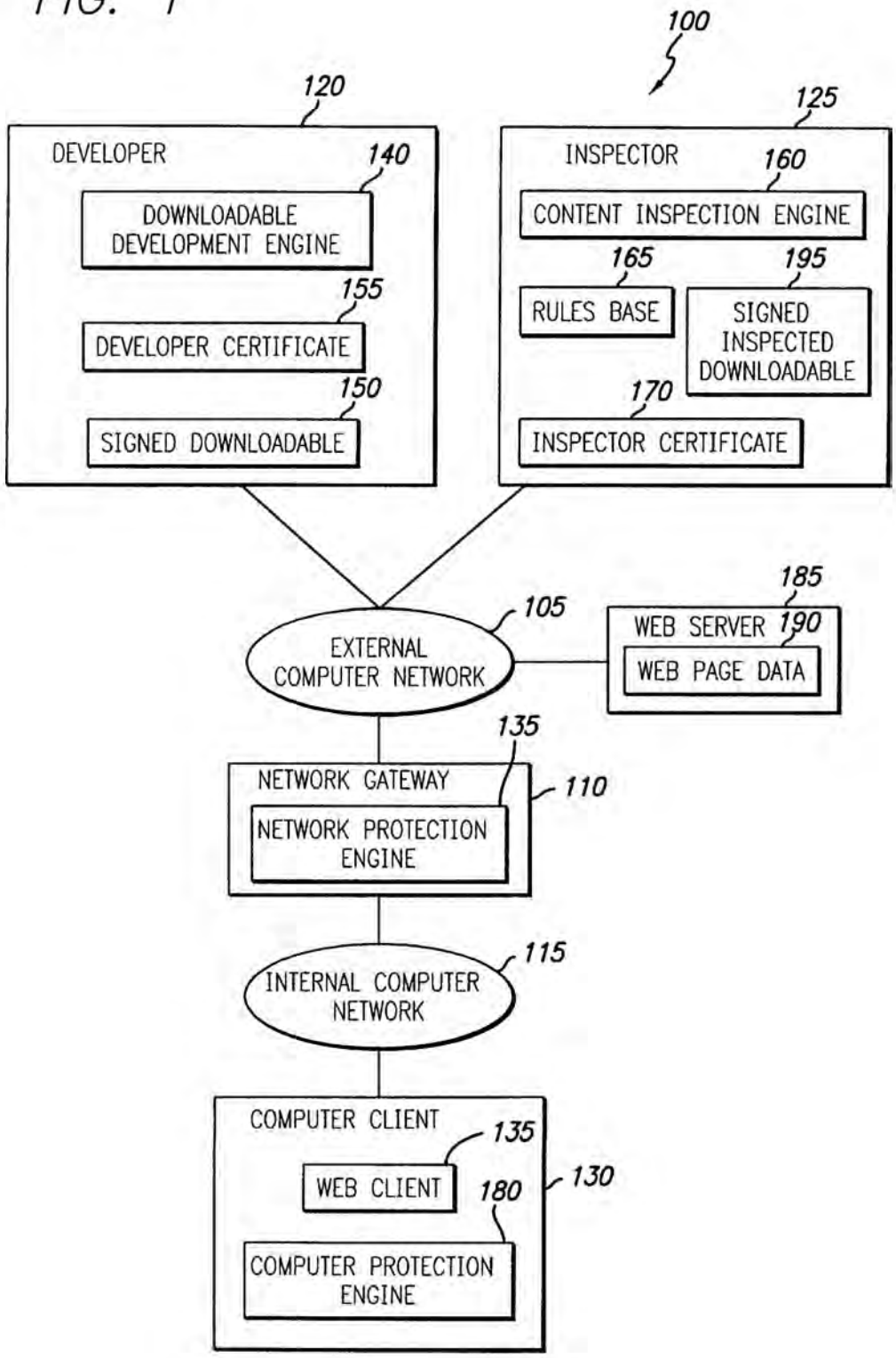
"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0" Las Vegas Convention Center/Pavillion 5 P5551, Nov. 18, 1996, 3 pages.

"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


100
120
DEVELOPER
140
DOWNLOADABLE DEVELOPMENT ENGINE
155
DEVELOPER CERTIFICATE
150
SIGNED DOWNLOADABLE
125
INSPECTOR
160
CONTENT INSPECTION ENGINE
165
RULES BASE
195
SIGNED INSPECTED DOWNLOADABLE
170
INSPECTOR CERTIFICATE
185
WEB SERVER
190
WEB PAGE DATA
105
EXTERNAL COMPUTER NETWORK
135
NETWORK GATEWAY
110
NETWORK PROTECTION ENGINE
115
INTERNAL COMPUTER NETWORK
COMPUTER CLIENT
135
WEB CLIENT
180
130
COMPUTER PROTECTION ENGINE

Case 3:17-cv-00183-BAS-NLS Document 1-1 Filed 07/01/16 PageID.42 Page 5 of 18

FIG. 2

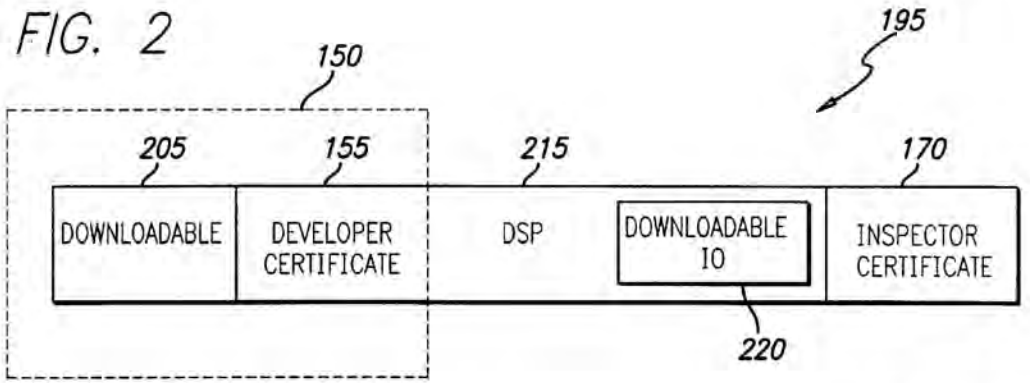


FIG. 3

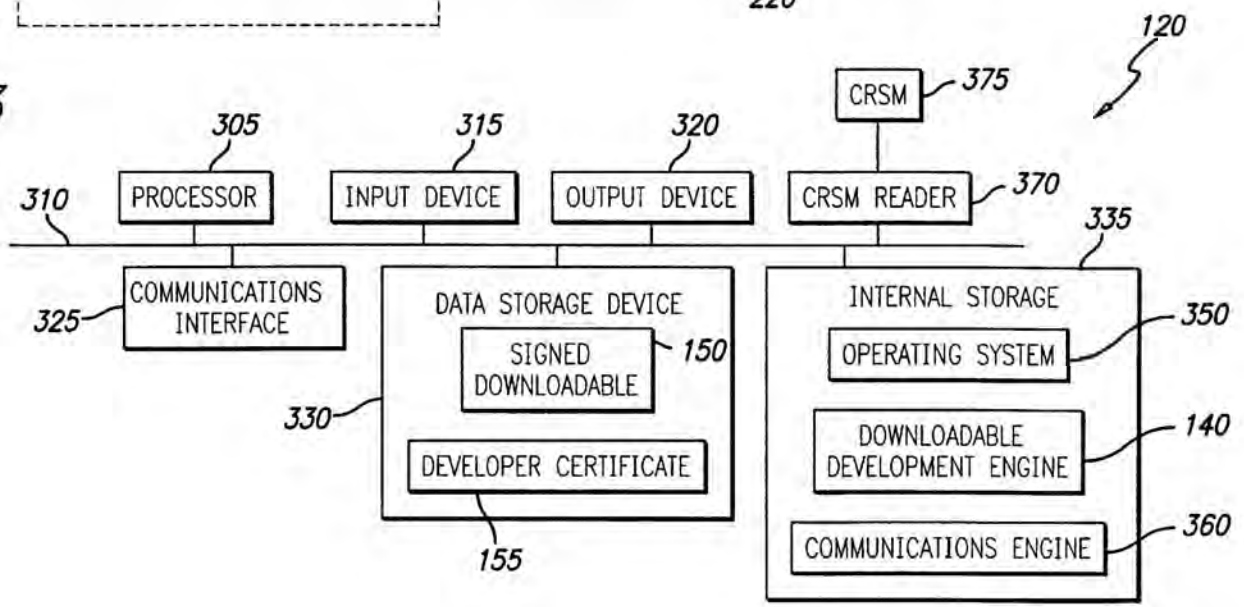
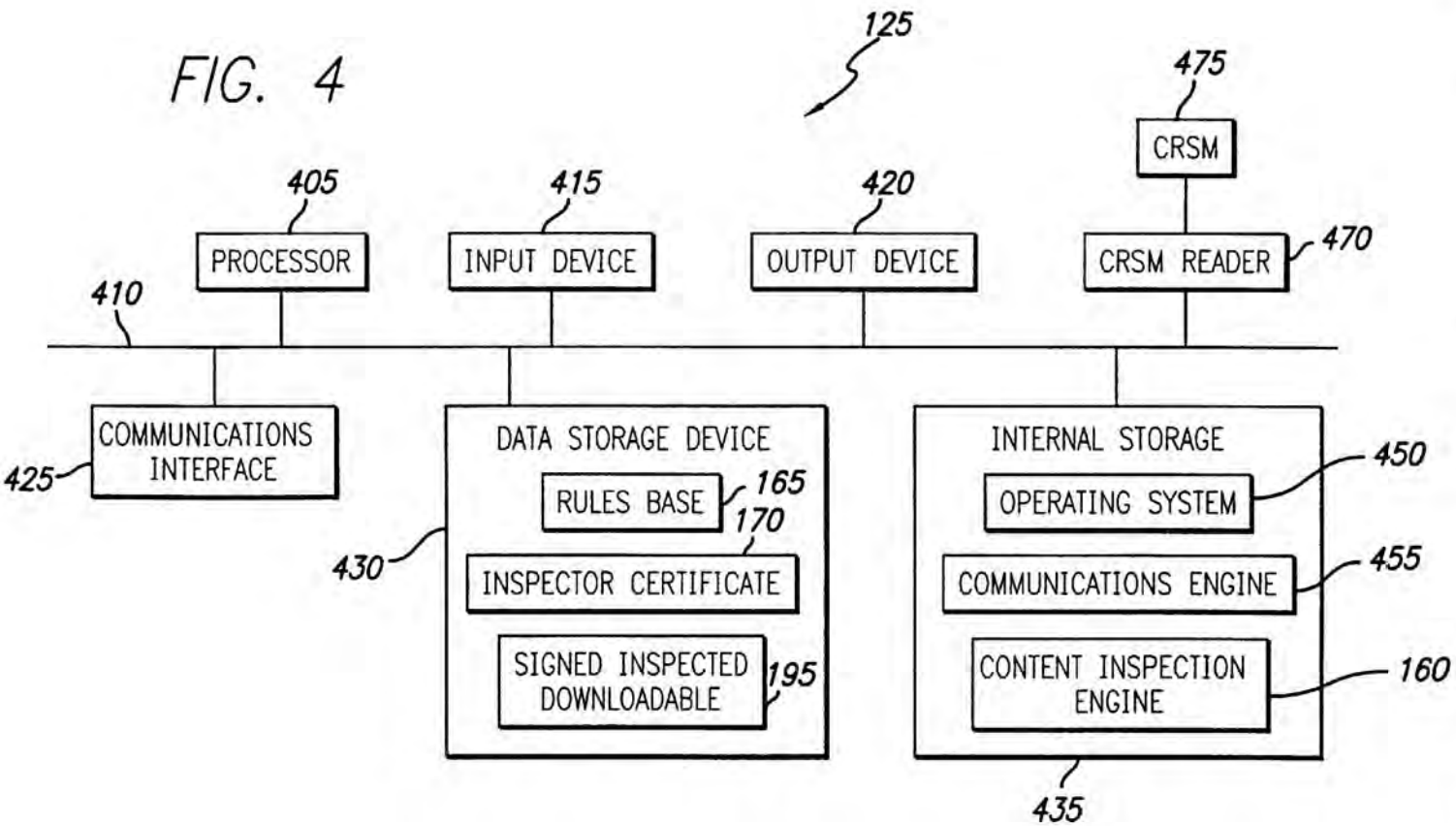


FIG. 4



Case 3:17-cv-00183-BAS-NLS Document 1-1 Filed 07/01/16 PageID.43 Page 6 of 18

FIG. 5

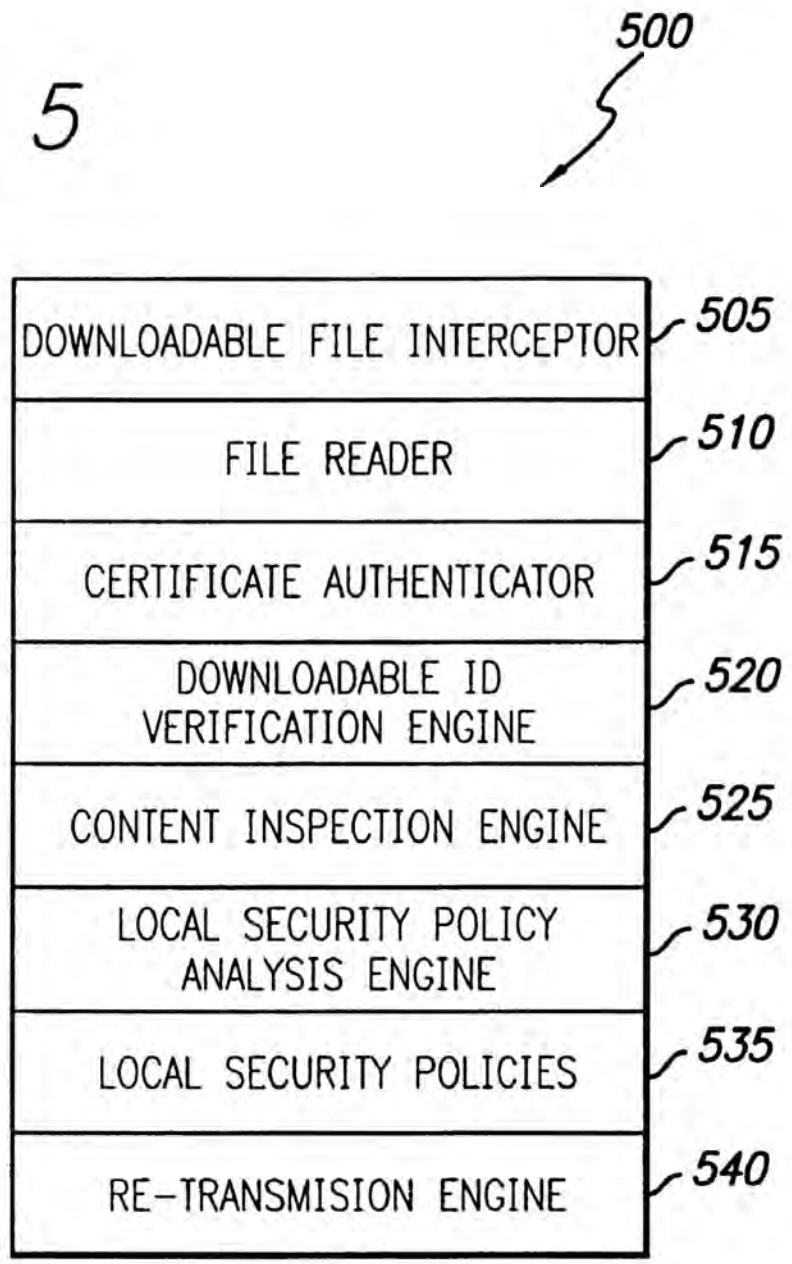

500
DOWNLOADABLE FILE INTERCEPTOR
505
FILE READER
510
CERTIFICATE AUTHENTICATOR
515
DOWNLOADABLE ID VERIFICATION ENGINE
520
CONTENT INSPECTION ENGINE
525
LOCAL SECURITY POLICY ANALYSIS ENGINE
530
LOCAL SECURITY POLICIES
535
RE-TRANSMISION ENGINE
540

FIG. 6

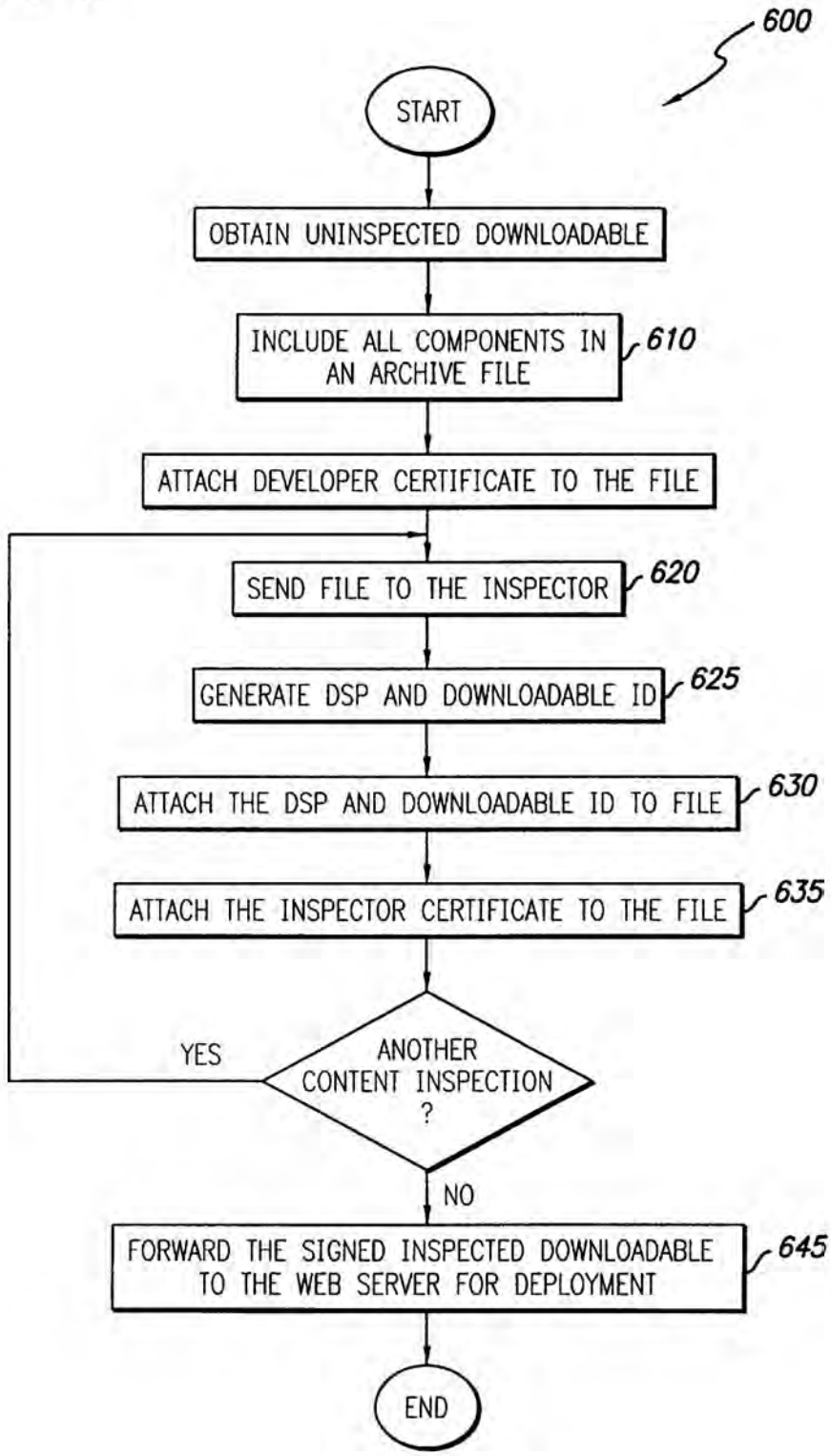
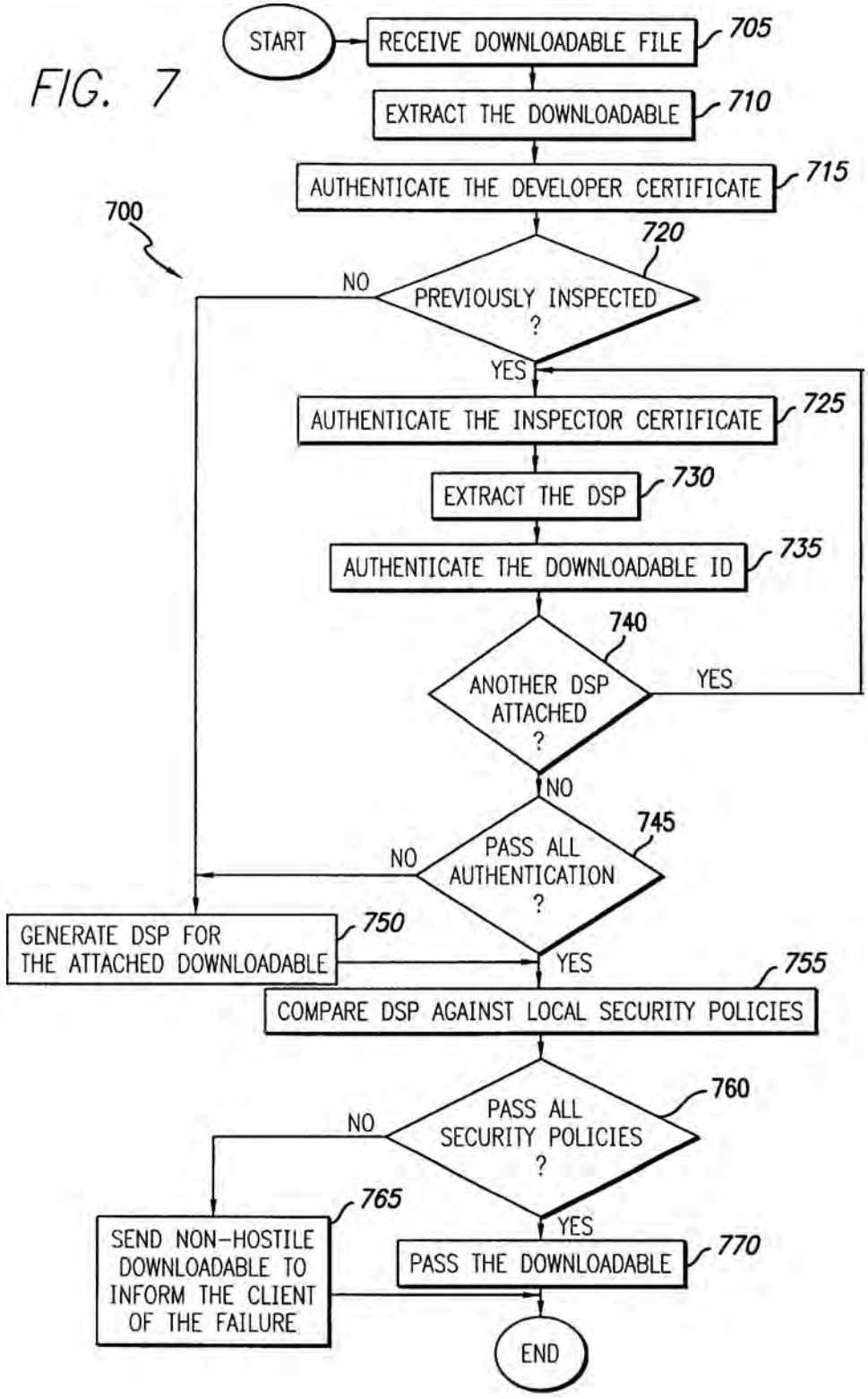
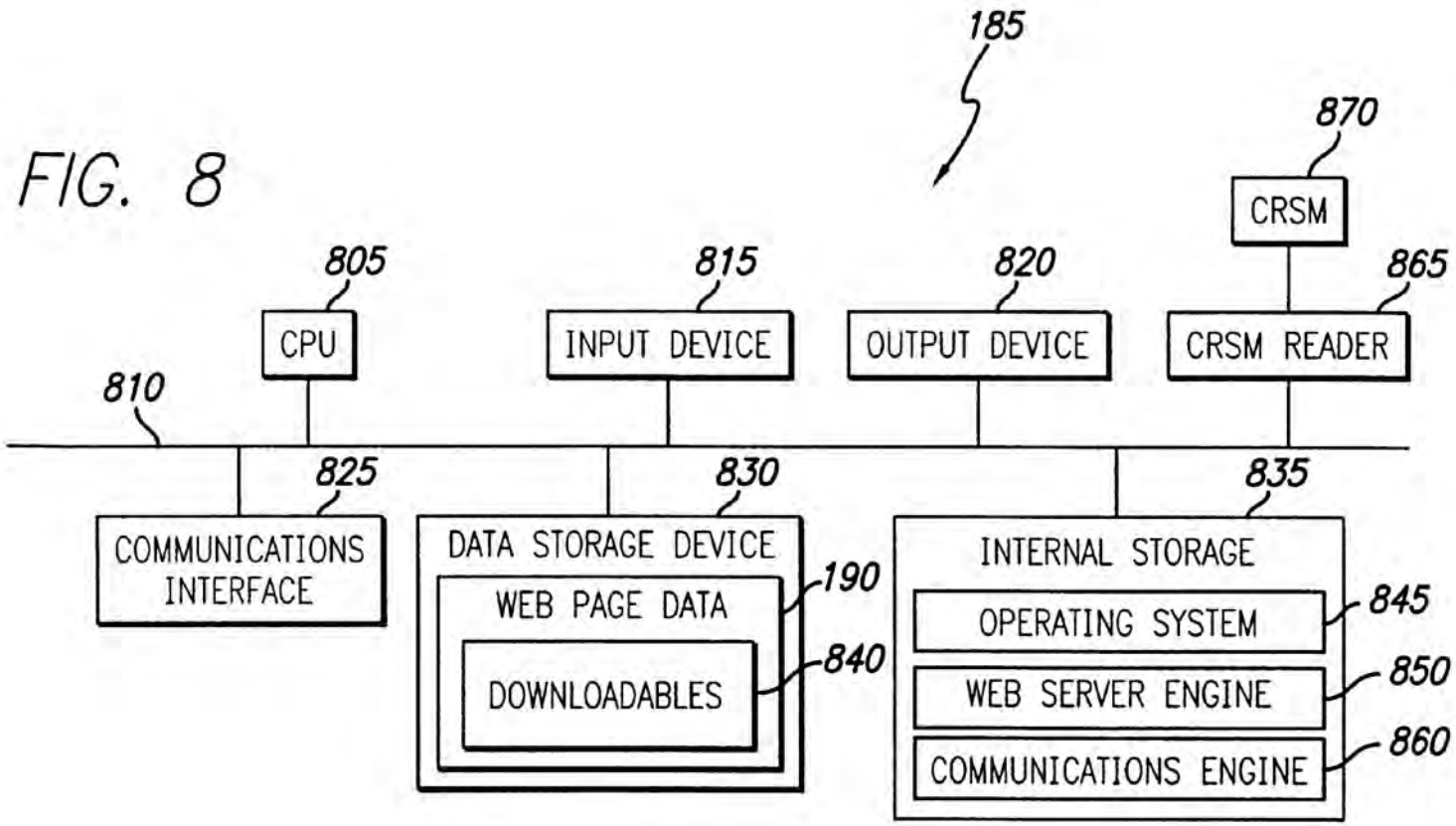

600
START
OBTAIN UNINSPECTED DOWNLOADABLE
INCLUDE ALL COMPONENTS IN AN ARCHIVE FILE
610
ATTACH DEVELOPER CERTIFICATE TO THE FILE
SEND FILE TO THE INSPECTOR
620
GENERATE DSP AND DOWNLOADABLE ID
625
ATTACH THE DSP AND DOWNLOADABLE ID TO FILE
630
ATTACH THE INSPECTOR CERTIFICATE TO THE FILE
635
YES
ANOTHER CONTENT INSPECTION ?
NO
FORWARD THE SIGNED INSPECTED DOWNLOADABLE TO THE WEB SERVER FOR DEPLOYMENT
645
END

FIG. 7


START
RECEIVE DOWNLOADABLE FILE
705
EXTRACT THE DOWNLOADABLE
710
AUTHENTICATE THE DEVELOPER CERTIFICATE
715
700
720
NO
PREVIOUSLY INSPECTED ?
YES
AUTHENTICATE THE INSPECTOR CERTIFICATE
725
EXTRACT THE DSP
730
AUTHENTICATE THE DOWNLOADABLE ID
735
740
ANOTHER DSP ATTACHED ?
YES
NO
745
PASS ALL AUTHENTICATION ?
NO
GENERATE DSP FOR THE ATTACHED DOWNLOADABLE
750
YES
755
COMPARE DSP AGAINST LOCAL SECURITY POLICIES
760
PASS ALL SECURITY POLICIES ?
NO
765
SEND NON-HOSTILE DOWNLOADABLE TO INFORM THE CLIENT OF THE FAILURE
YES
PASS THE DOWNLOADABLE
770
END

Case 3:17-cv-00183-BAS-NLS Document 1-1 Filed 07/01/16 PageID.47 Page 10 of 18

FIG. 8



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 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 fall-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

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 developer'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

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 an 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 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.

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.

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.

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

Exhibit B

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.**[7] **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–archive/summer96/0338.html, p. 1–2.*

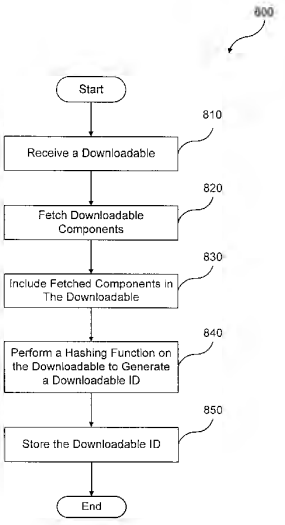
(List continued on next page.)

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



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 Microsfot ActiveX Development Kit", Aug. 13, 1996, 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, 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?se . . . 2ehts%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 Softwre 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 Consulant: 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

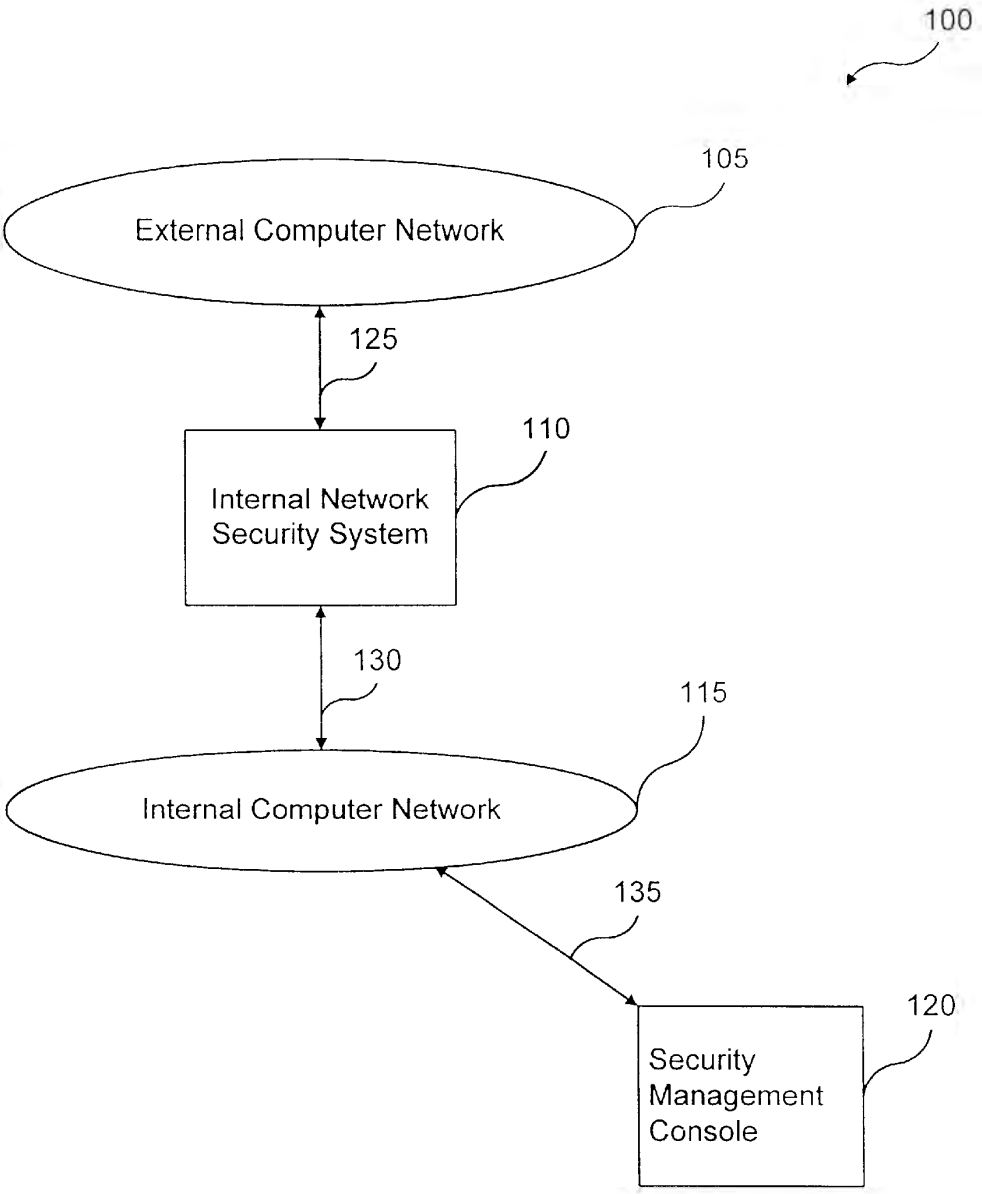

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

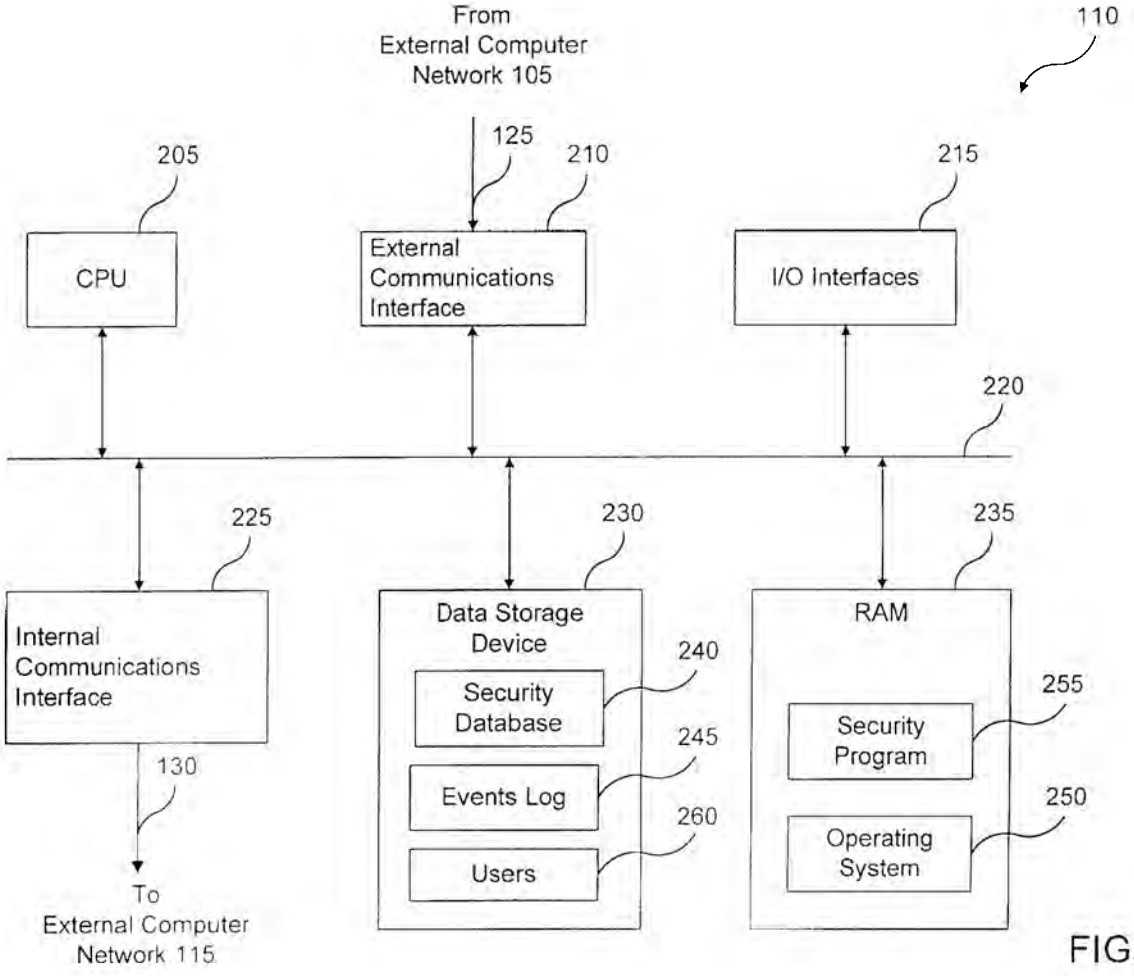

From
External Computer
Network 105
110
125
205
210
215
CPU
External
Communications
Interface
I/O Interfaces
220
225
230
235
Internal
Communications
Interface
Data Storage
Device
RAM
240
Security
Database
255
Security
Program
245
Events Log
250
Operating
System
260
Users
130
To
External Computer
Network 115

FIG. 2

Case 3:17-cv-00183-BAS-NLS Document 1-2 Filed 07/01/16 PageID.61 Page 6 of 19

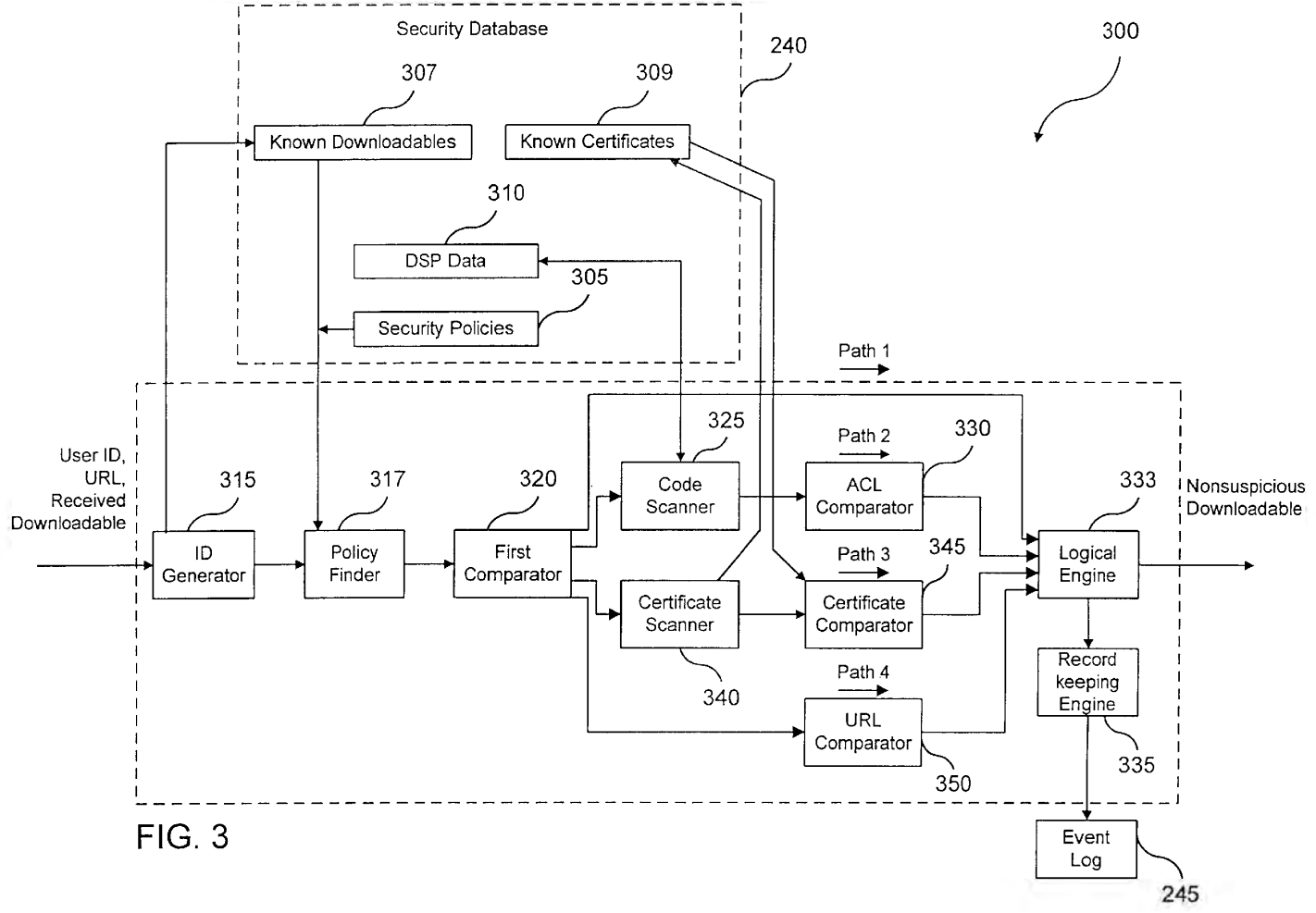


FIG. 3

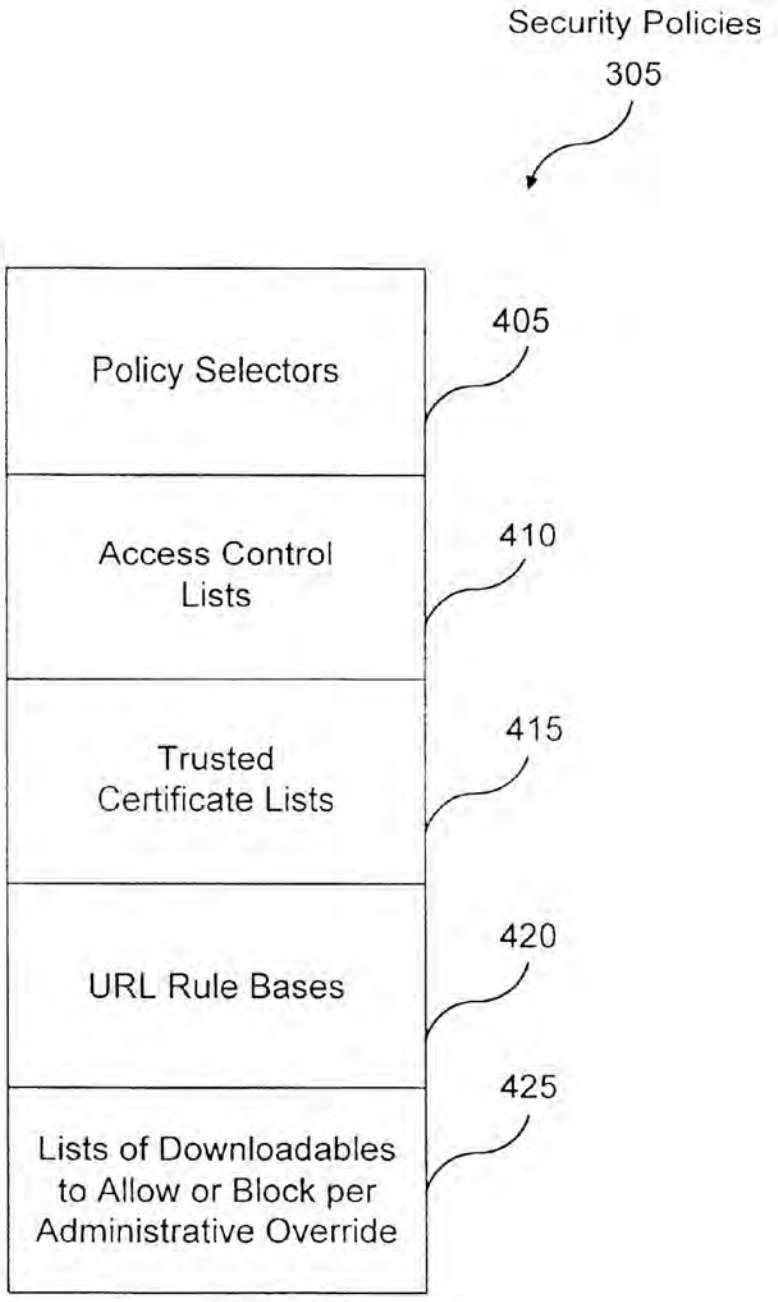

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

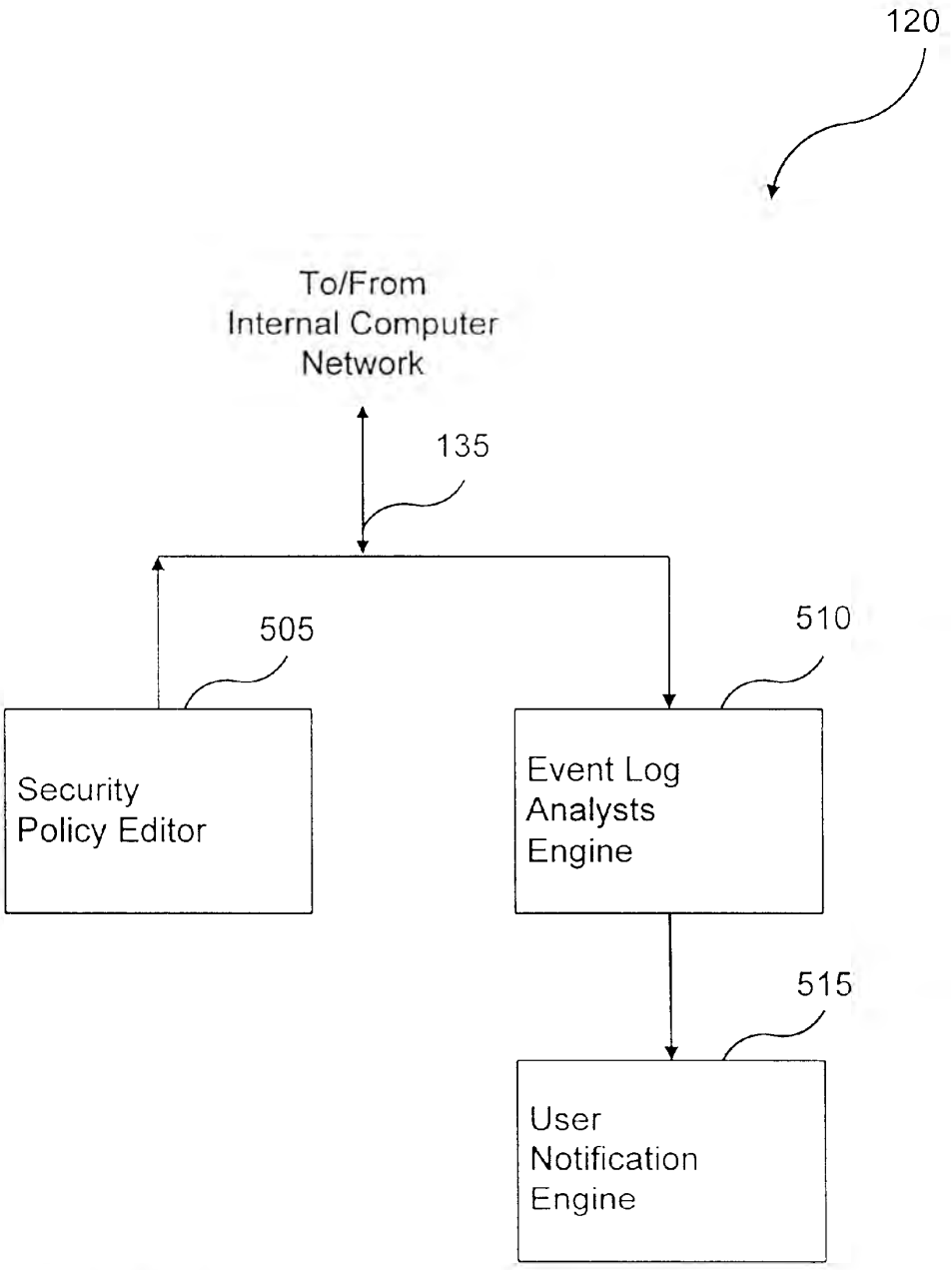

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

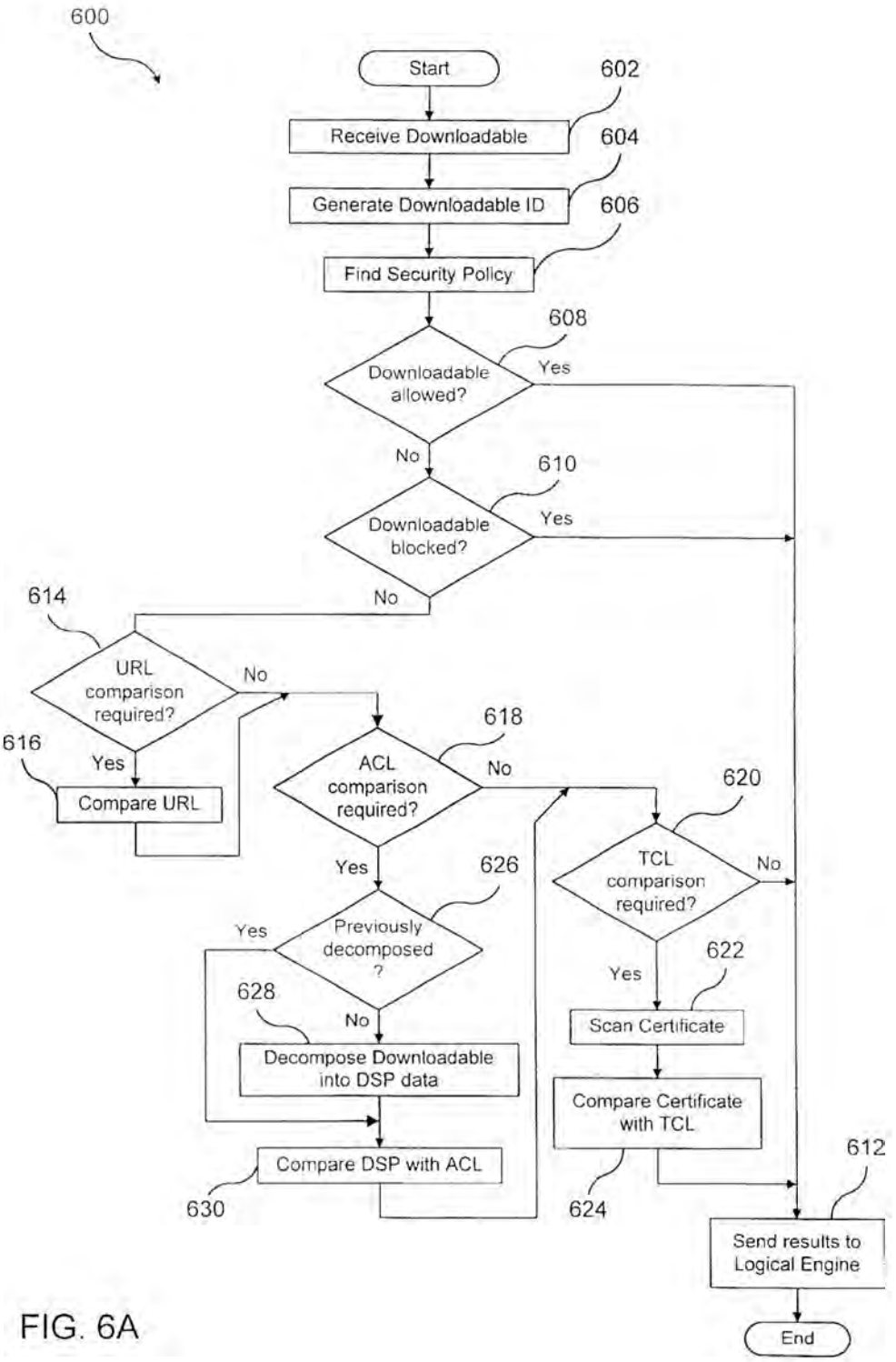

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
626
Previously decomposed ?
Yes
628
No
Decompose Downloadable into DSP data
Compare DSP with ACL
630
Yes
622
Scan Certificate
Compare Certificate with TCL
624
612
Send results to Logical Engine
End

FIG. 6A

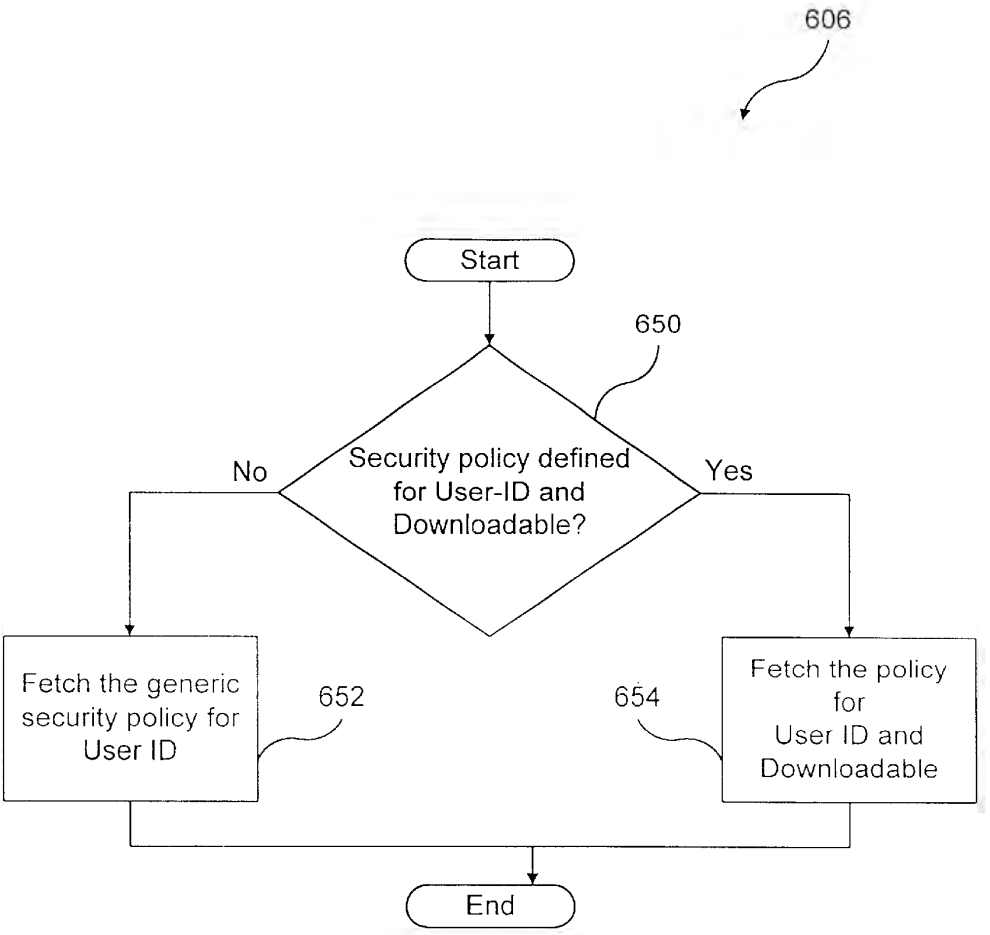

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

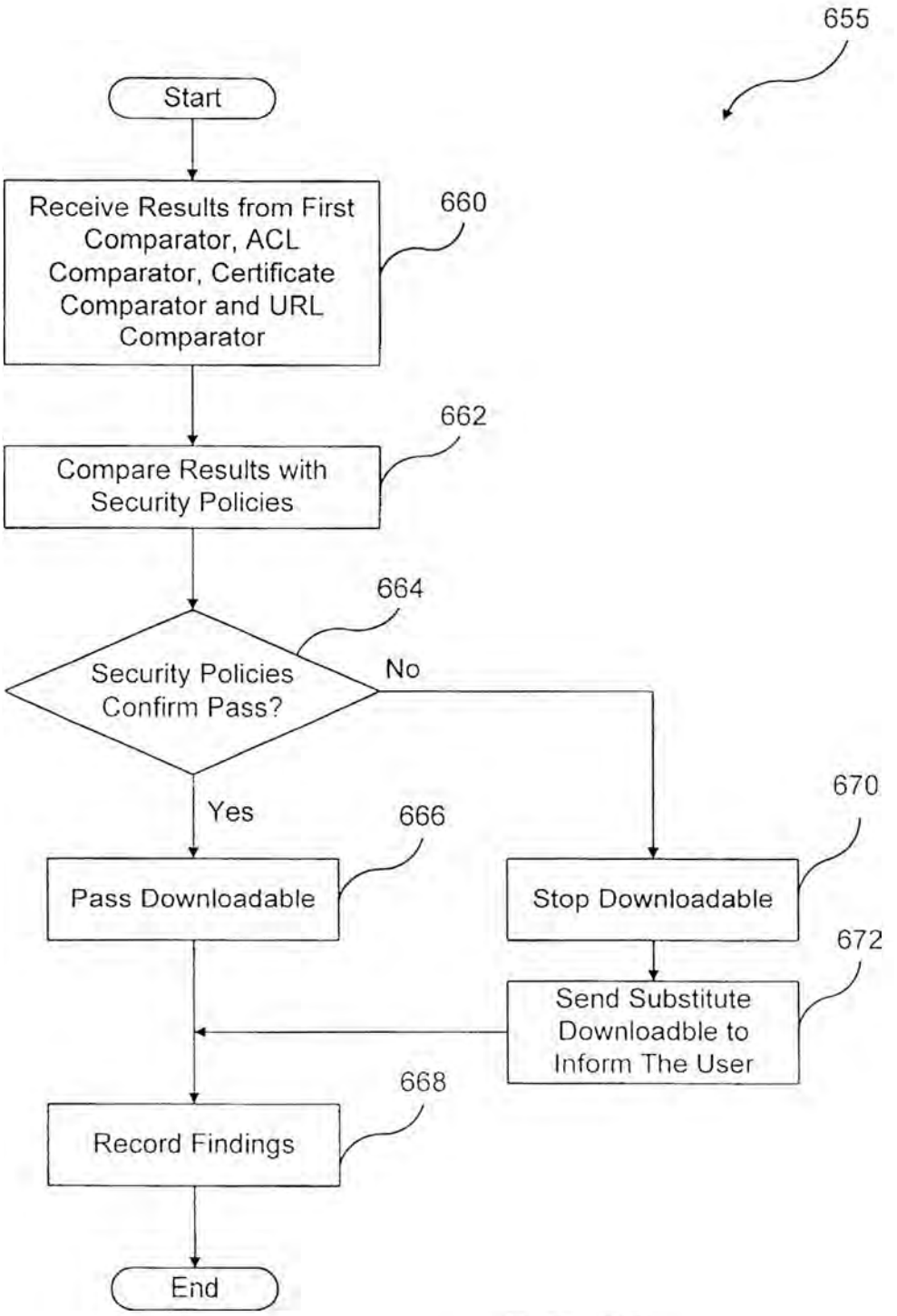

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings
668
End

FIG. 6C

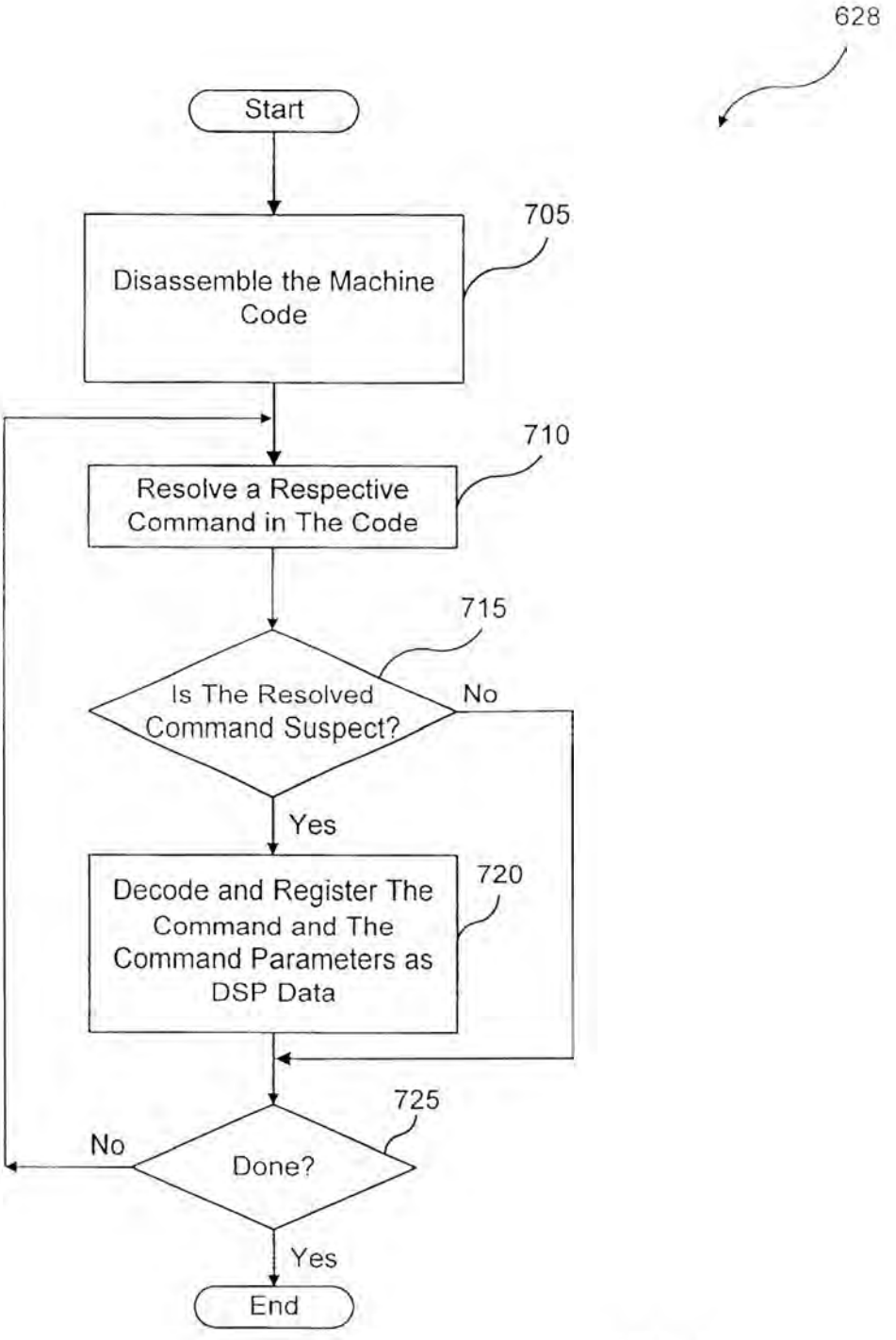

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

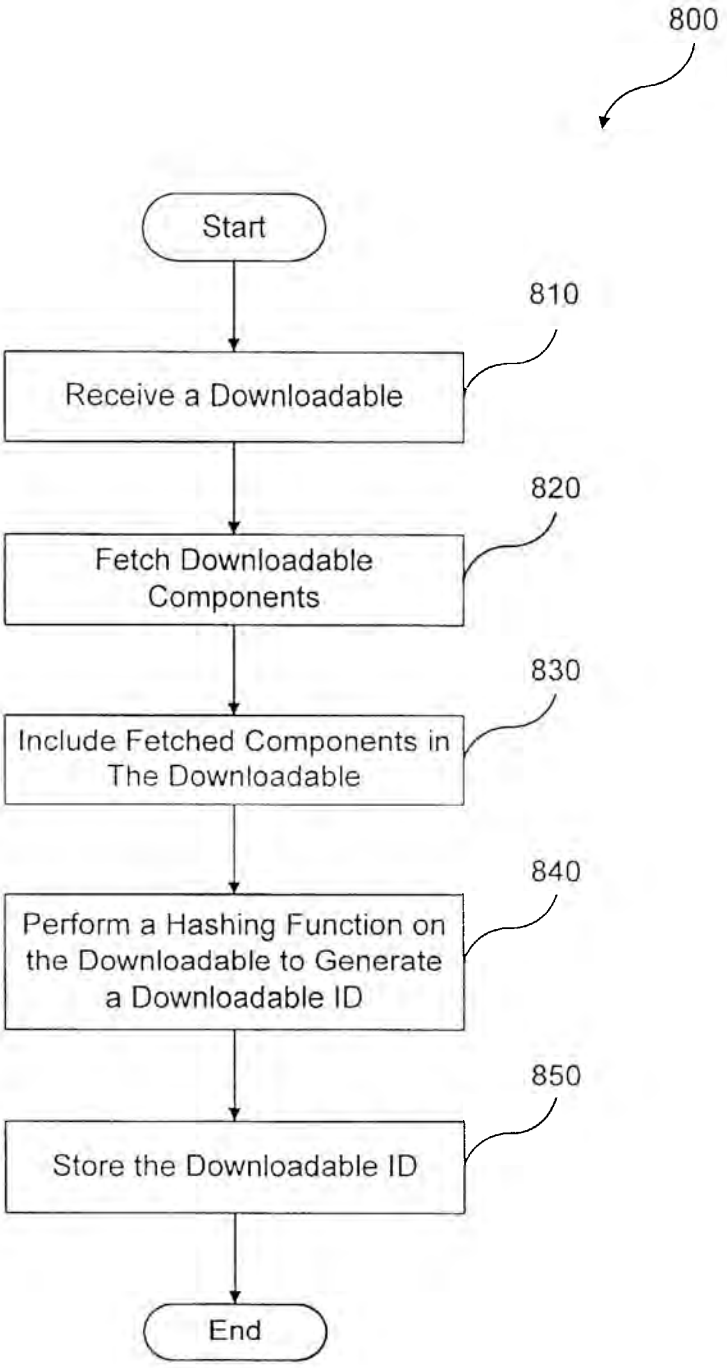

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

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

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

FIG. **6**C 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

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 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

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 path2 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.

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

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 came 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** 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**.

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 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.

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

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;

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.

* * * * *

Exhibit D

US008079086B1

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

(10) **Patent No.: US 8,079,086 B1**
(45) **Date of Patent: *Dec. 13, 2011**

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

(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)

(73) Assignee: **Finjan, Inc.**, San Jose, CA (US)

(*) 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.: **12/471,942**

(22) Filed: **May 26, 2009**

Related U.S. Application Data

(63) Continuation of application No. 11/370,114, filed on Mar. 7, 2006, now Pat. No. 7,613,926, which is a continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, which is 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.

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

(51) **Int. Cl.**

G06F 21/00	(2006.01)
G06F 11/30	(2006.01)
G06F 15/16	(2006.01)
H04L 9/32	(2006.01)

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

(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
5,359,659 A	10/1994	Rosenthal	726/24
5,361,359 A	11/1994	Tajalli et al.	726/23

(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.

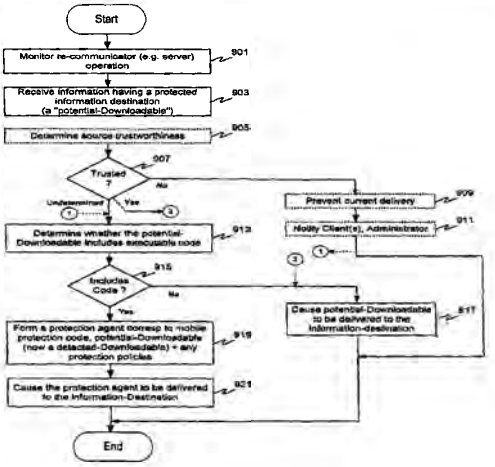
(Continued)

Primary Examiner — Christopher Revak
(74) *Attorney, Agent, or Firm* — Dawn-Marie Bey; 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 TN 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 "recommunicator," 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.

42 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

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	Edery et al.	726/22
7,143,444	B2	11/2006	Porras et al.	726/30
7,210,041	B1	4/2007	Gryaznov et al.	713/188
7,308,648	B1	12/2007	Buchthal et al.	715/234
7,343,604	B2	3/2008	Grabarnik et al.	719/313
7,418,731	B2	8/2008	Touboul	726/22
7,613,926	B2 *	11/2009	Edery et al.	713/181
7,647,633	B2 *	1/2010	Edery et al.	726/22
2003/0014662	A1	1/2003	Gupta et al.	726/23
2003/0101358	A1	5/2003	Porras et al.	726/4
2004/0073811	A1	4/2004	Sanin	726/13
2004/0088425	A1	5/2004	Rubinstein et al.	709/230
2005/0050338	A1	3/2005	Liang et al.	713/188
2005/0172338	A1	8/2005	Sandu et al.	726/22
2006/0031207	A1	2/2006	Bjarnestam et al.	707/3
2006/0048224	A1	3/2006	Duncan et al.	726/22
2008/0066160	A1	3/2008	Becker et al.	726/4
2010/0195909	A1	8/2010	Wasson et al.	382/176

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 18th 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, 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, Surfin Shield™," Article published on the Internet by Finjan Software Ltd., 2 pp. 1996.

Finjan Softrware 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.
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 Magaine*, 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.
Power, James, "Notes on Formal Language Theory and Parsing," National University of Ireland, pp. 1-40, 1999.

* cited by examiner

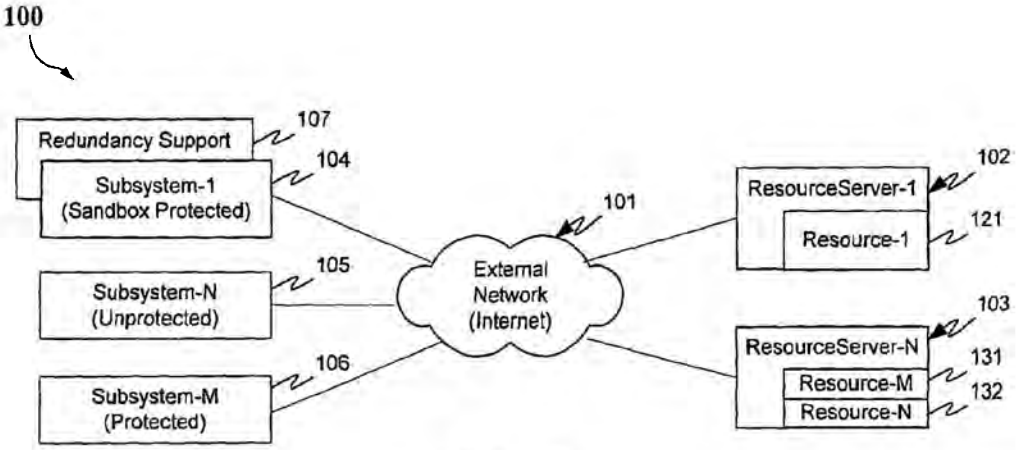


FIG. 1a

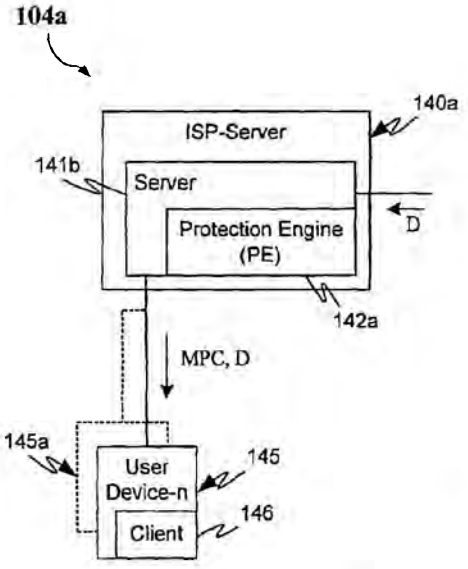


FIG. 1b

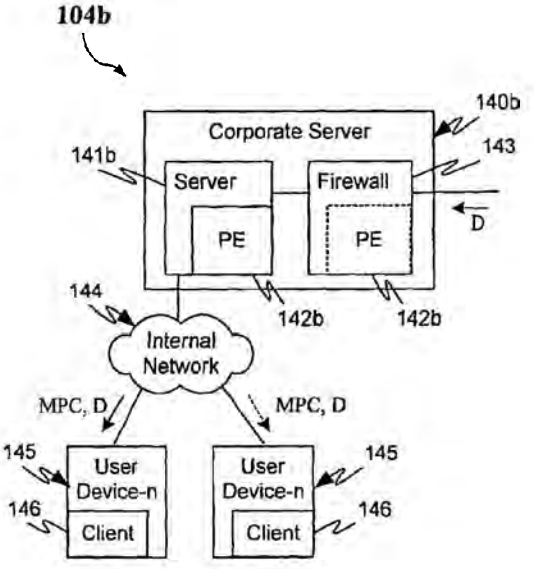


FIG. 1c

Case 3:17-cv-00183-BAS-NLS Document 1-4 Filed 07/01/16 PageID.114 Page 6 of 26

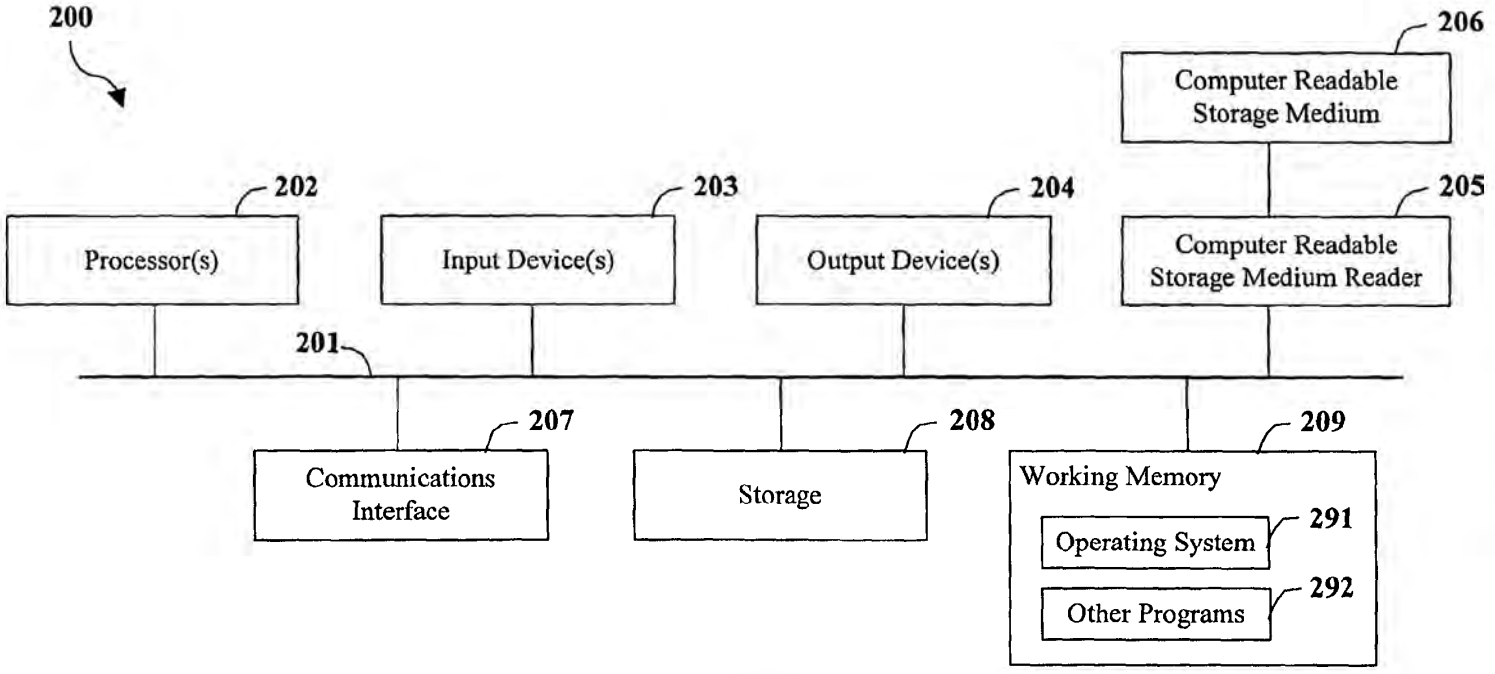


FIG. 2

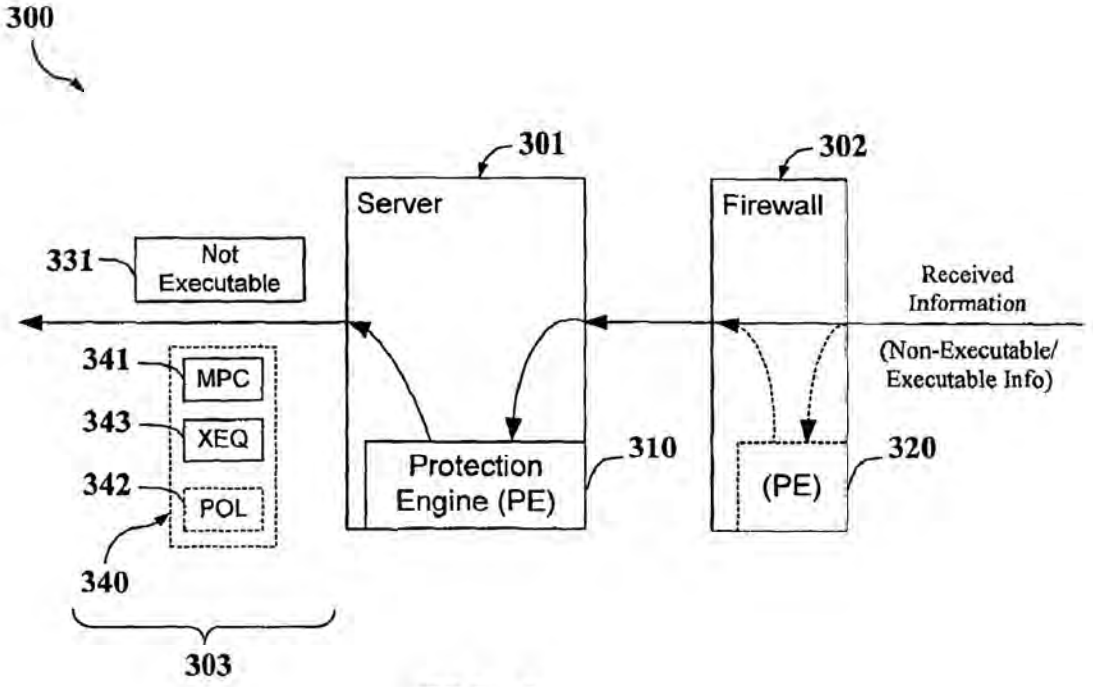

300
301
Server
302
Firewall
331
Not Executable
Received Information
(Non-Executable/ Executable Info)
341
MPC
343
XEQ
342
POL
340
303
Protection Engine (PE)
310
(PE)
320

FIG. 3

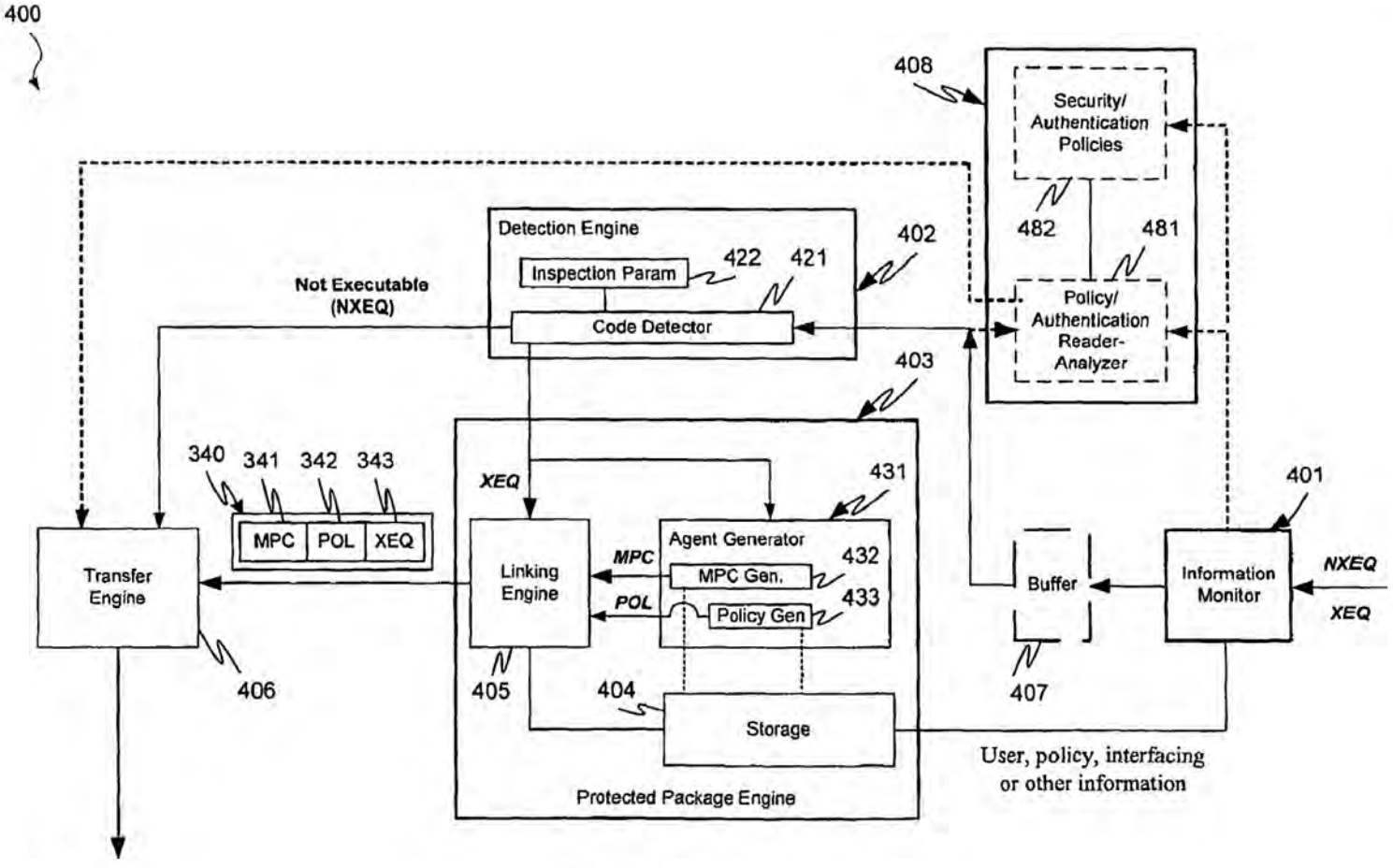

400
408
Security/ Authentication Policies
482
481
Policy/ Authentication Reader-Analyzer
Detection Engine
402
Inspection Param
422
421
Code Detector
Not Executable (NXEQ)
403
340
341
342
343
MPC
POL
XEQ
XEQ
431
Agent Generator
MPC
MPC Gen.
432
POL
Policy Gen
433
Transfer Engine
Linking Engine
401
Buffer
Information Monitor
NXEQ
XEQ
406
405
404
407
Storage
User, policy, interfacing or other information
Protected Package Engine

FIG. 4

Case 3:17-cv-00183-BAS-NLS Document 1-4 Filed 07/01/16 PageID.117 Page 9 of 26

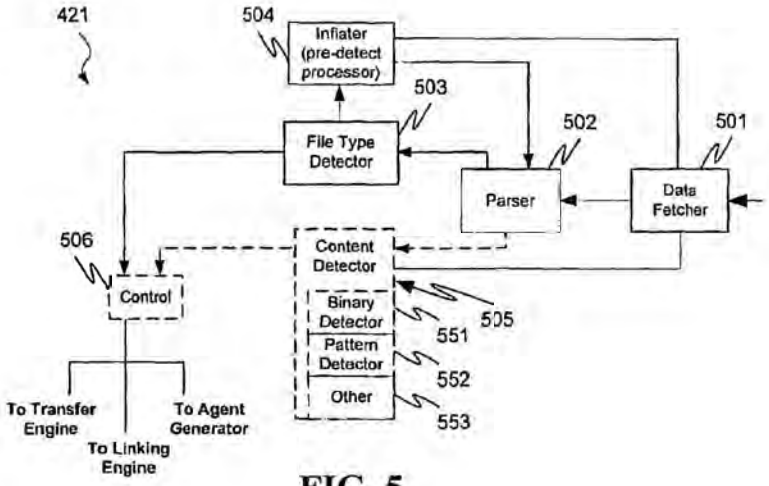


FIG. 5

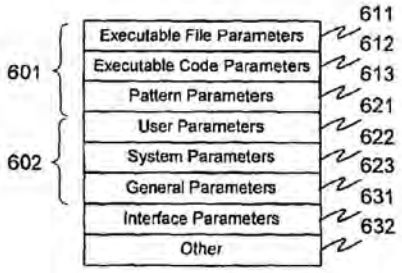


FIG. 6a

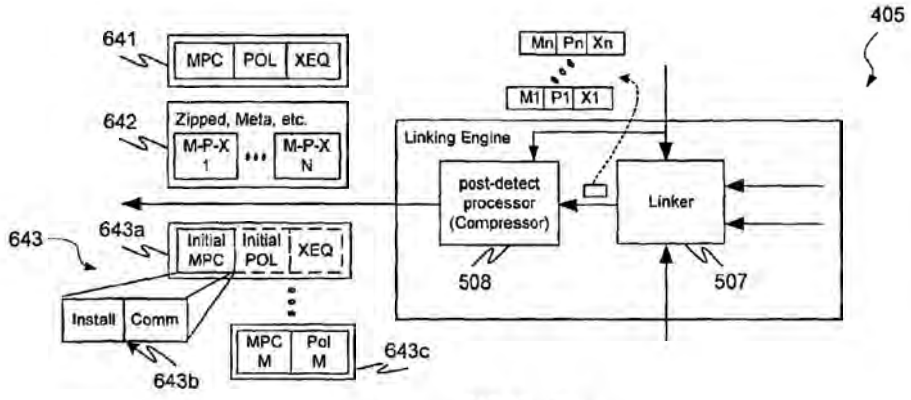


FIG. 6b

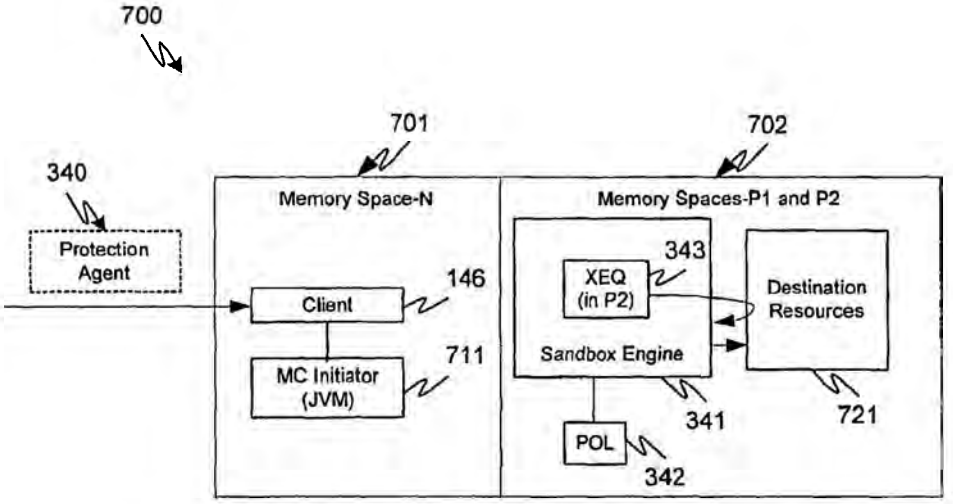


FIG. 7a

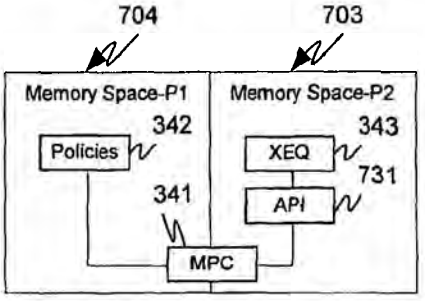


FIG. 7b

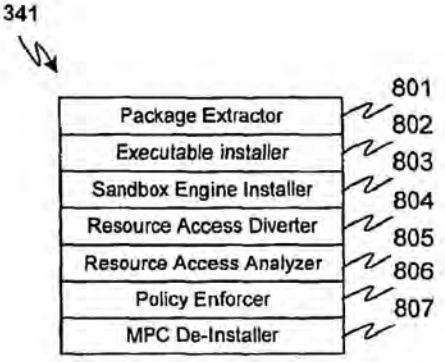


FIG. 8

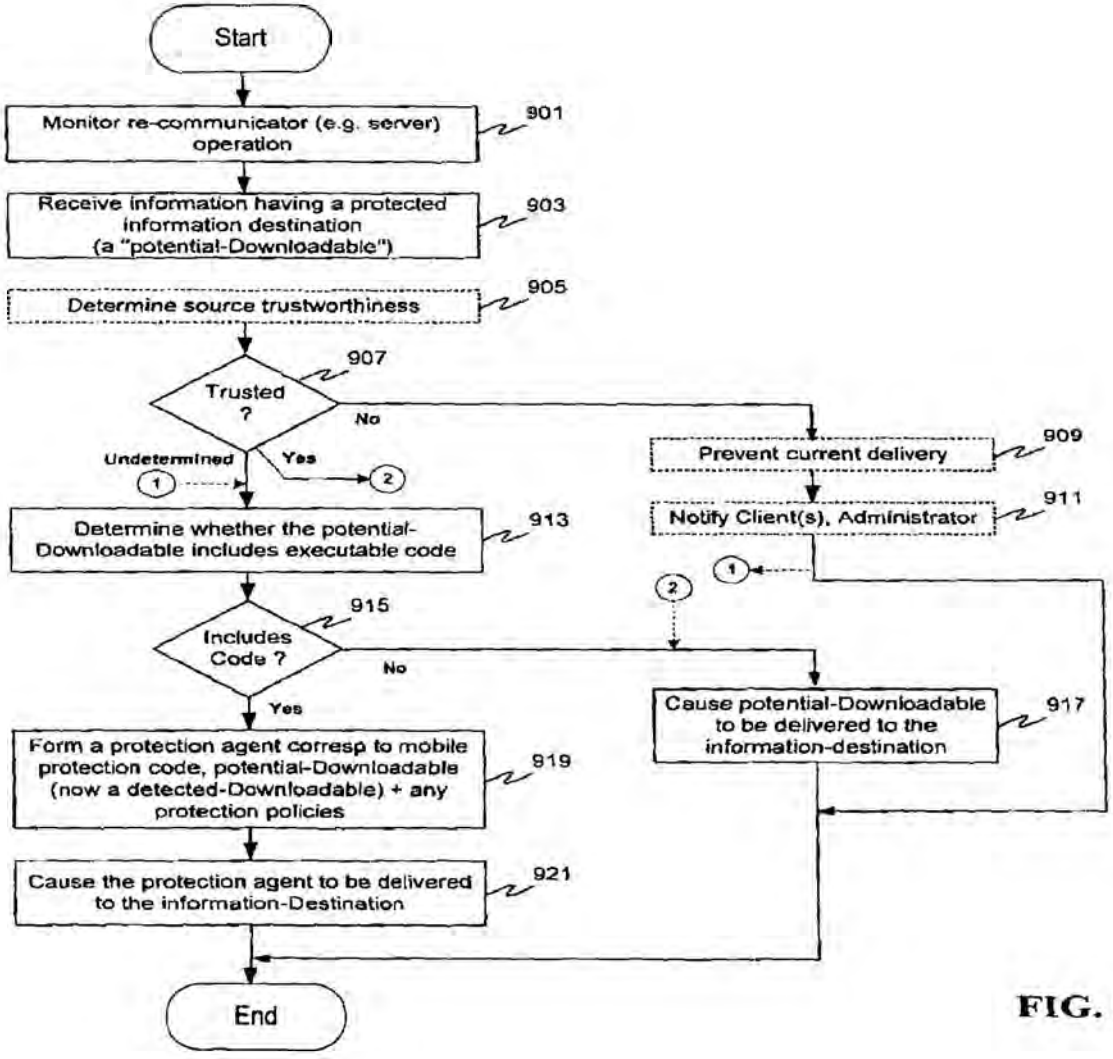

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
907
Trusted ?
No
Undetermined
Yes
1
2
Prevent current delivery
909
Notify Client(s), Administrator
911
Determine whether the potential-Downloadable includes executable code
913
915
Includes Code ?
No
Yes
Cause potential-Downloadable to be delivered to the information-destination
917
Form a protection agent corresp to mobile protection code, potential-Downloadable (now a detected-Downloadable) + any protection policies
919
Cause the protection agent to be delivered to the information-Destination
921
End

FIG. 9

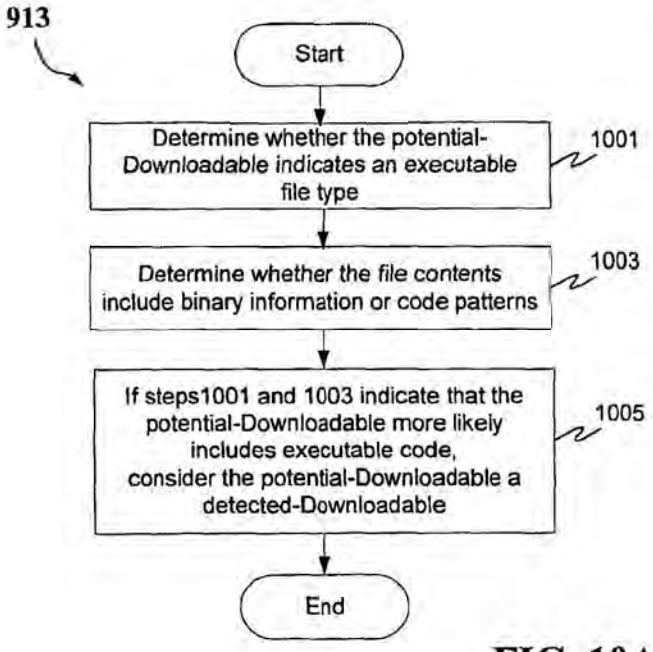


FIG. 10A

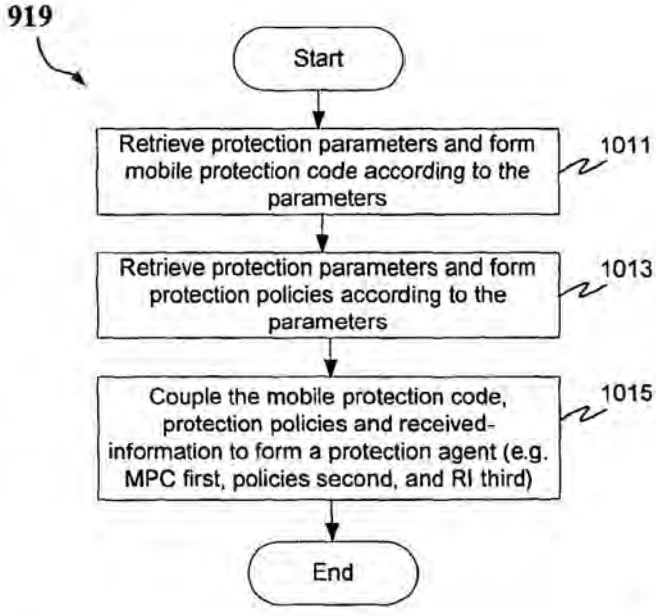


FIG. 10B

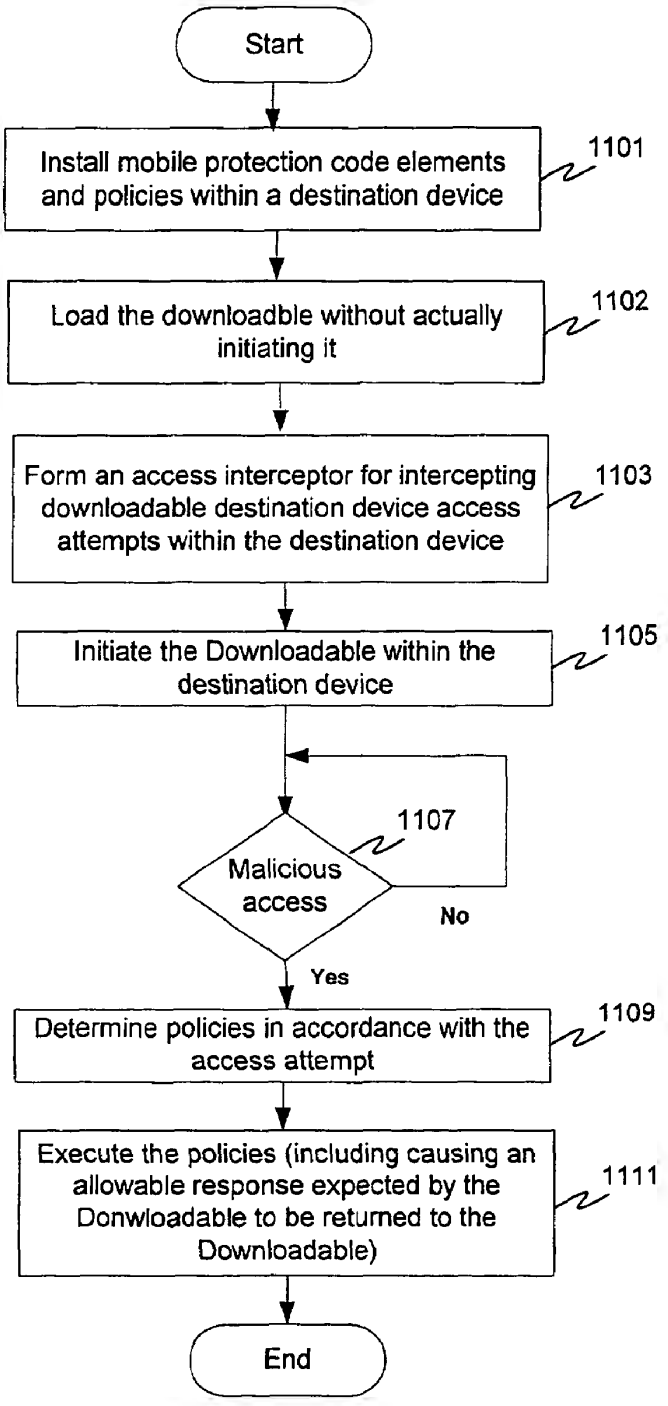

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable within the destination device
1105
Malicious access
1107
No
Yes
Determine policies in accordance with the access attempt
1109
Execute the policies (including causing an allowable response expected by the Donwloadable to be returned to the Downloadable)
1111
End

FIG. 11

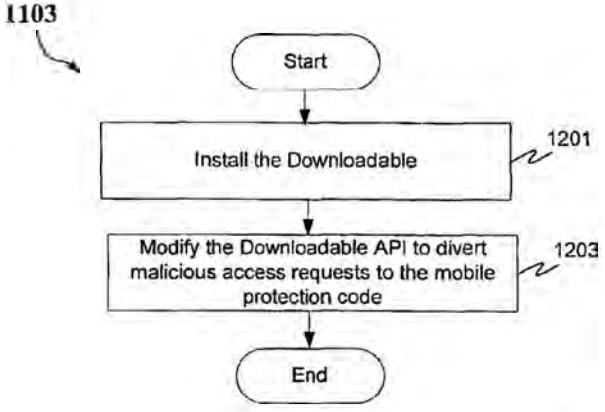


FIG. 12a

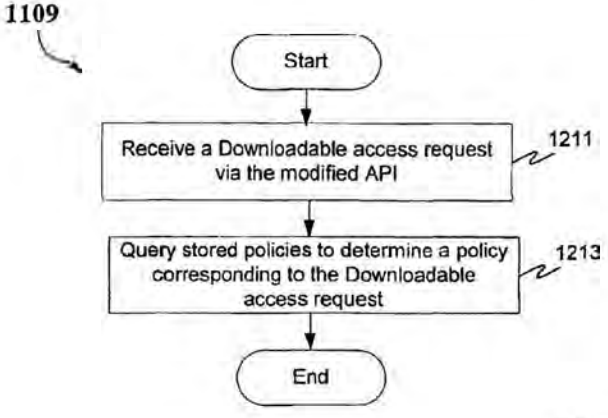


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application is a continuation of assignee's application Ser. No. 11/370,114, filed Mar. 7, 2006, now U.S. Pat. No. 7,613,926, entitled "Method and System for Protecting a Computer and a Network from Hostile Downloadables," which is a continuation of 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", all of which are 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 U.S. patent application 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 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-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 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 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. **1***a* is a block diagram illustrating a network system in accordance with an embodiment of the present invention;

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

FIG. **1***c* 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 ru 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. **6***a* is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. **6***b* 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. **7***a* is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. **7***b* is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. **7***a*, 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. **10***a* 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. **10***b* 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. **12***a* is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. **12***b* 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. **1***a* through **1***c* illustrate a computer network system **100** according to an embodiment of the invention. FIG. **1***a* broadly illustrates system **100**, while FIGS. **1***b* and **12** of **1***c* illustrate exemplary protectable subsystem implementations corresponding with system **104** or **106** of FIG. **1***a*.

Beginning with FIG. **1***a*, 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. **1***a* 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. **1***b* and **1***c* 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** band firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements **11**) 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, 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 ~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, OSL, 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 **110** 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 exten-

sibly 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 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 NM 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 (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 me 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 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 **20** (e.g. see FIG. **6***b*). 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 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. (Anyone 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**, inflator **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 as 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 inflator **504**. File inflator **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, inflator **504** returns such information to parser **502**. File inflator **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. **6***b*). 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 **643***b* of FIG. **6***b*. (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 (**643***a* of FIG. **6***b*). 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 **643***c*, 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. **6***a* 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. **6***b* 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 concatination 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. **7***a*, **7***b* and **8** illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. **7***a*, 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 IA T can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IA T 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.

Turning to FIG. **8** with reference to FIG. **7***b*, 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 as/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 **20** 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. **10***a* 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. **10***a*, 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. **10***b* 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. **12***a* 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 ill, 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. **12***b* 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;

deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable;

appending a representation of the Downloadable security profile data to the 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 appended Downloadable includes a digital certificate.

8. The computer-based method of claim 1 wherein said deriving Downloadable security profile data comprises disassembling the incoming Downloadable.

9. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable;

a Downloadable scanner coupled with said receiver for deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable;

a file appender coupled with said Downloadable scanner, for appending a representation of the Downloadable security profile data to the Downloadable, to generate an appended Downloadable; and

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

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

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

12. The system of claim 9 wherein the Downloadable includes program script.

13. The system of claim 9 wherein suspicious computer operations include calls made to an operating system, a file system, a network system, and to memory.

14. The system of claim 9 wherein the Downloadable security profile data includes a URL from where the Downloadable originated.

15. The system of claim 9 wherein the appended Downloadable includes a digital certificate.

16. The system of claim 9 wherein said Downloadable scanner comprises a disassembler for disassembling the incoming Downloadable.

17. A computer-based method, comprising the steps of:

receiving an incoming Downloadable;

deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and

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

18. The computer-based method of claim 17 wherein the transport protocol is an application transport protocol, and wherein the Downloadable security profile data is inserted as a header within the transport protocol transmission.

19. The computer-based method of claim 18 wherein the application transport protocol is HTTP.

20. The computer-based method of claim 18 wherein the application transport protocol is FTP.

21. The computer-based method of claim 17 wherein the transport protocol is a network transport protocol, and wherein the Downloadable security profile data is inserted as a frame within the transport protocol transmission.

22. The computer-based method of claim 21 wherein the network transport protocol is TCP/IP.

23. The computer-based method of claim 21 wherein the network transport protocol is UDP.

24. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable;

a Downloadable scanner coupled with said receiver, for deriving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and

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

25. The system of claim 24 wherein the transport protocol is an application transport protocol and wherein the Downloadable security profile data is inserted as a header within the transport protocol transmission.

26. The system of claim 25 wherein the application transport protocol is HTTP.

27. The system of claim 25 wherein the application transport protocol is FTP.

28. The system of claim 24 wherein the transport protocol is a network transport protocol, and wherein the Downloadable security profile data is inserted as a frame within the transport protocol transmission.

29. The system of claim 28 wherein the network transport protocol is TCP/IP.

30. The system of claim 28 wherein the network transport protocol is UDP.

31. A computer-based method, comprising the steps of:

receiving an incoming Downloadable;

receiving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable;

appending a representation of the Downloadable security profile data to the Downloadable, to generate an appended Downloadable; and

transmitting the appended Downloadable to a destination computer.

32. The computer-based method of claim 31 further comprising forwarding the Downloadable to an external computer, for deriving the Downloadable security profile data.

33. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable, and for receiving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable;

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

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

34. The system of claim 33 wherein said transmitter forwards the Downloadable to an external computer, for deriving the Downloadable security profile data, and wherein said receiver receives the security profile data from the external computer.

35. A computer-based method, comprising the steps of:

receiving an incoming Downloadable;

receiving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and

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

36. The computer-based method of claim 35 further comprising forwarding the Downloadable to an external computer, for deriving the Downloadable security profile data.

37. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable, and for receiving security profile data for the Downloadable, including a list of suspicious computer operations that may be attempted by the Downloadable; and

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

38. The system of claim 37 wherein said transmitter forwards the Downloadable to an external computer, for deriving the Downloadable security profile data, and wherein said receiver receives the security profile data from the external computer.

39. A computer-based method, comprising the steps of:

receiving an incoming Downloadable;

retrieving security profile data for the incoming Downloadable from a database of Downloadable security profiles indexed according to Downloadable IDs, based on an ID of the incoming Downloadable, the security profile data 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.

40. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable;

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 an ID of the incoming Downloadable, 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 a representation 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.

41. A computer-based method, comprising the steps of:

receiving an incoming Downloadable;

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

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

42. A system for managing Downloadables, comprising:

a receiver for receiving an incoming Downloadable;

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 an ID of the incoming Downloadable, 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.

* * * * *

Exhibit E

US009189621B2

(12) **United States Patent**
Touboul

(10) **Patent No.: US 9,189,621 B2**
(45) **Date of Patent: *Nov. 17, 2015**

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

(71) Applicant: **Finjan, Inc.**, East Palo Alto, CA (US)

(72) Inventor: **Shlomo Touboul**, Kefar-Haim (IL)

(73) Assignee: **Finjan, Inc.**, East Palo Alto, CA (US)

(*) 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.: **14/619,363**

(22) Filed: **Feb. 11, 2015**

(65) **Prior Publication Data**

US 2015/0180885 A1 Jun. 25, 2015

Related U.S. Application Data

(63) Continuation of application No. 14/155,835, filed on Jan. 15, 2014, which is a continuation of application No. 13/290,708, filed on Nov. 7, 2011, now Pat. No. 8,677,494, which is a continuation of application No.

(Continued)

(51) **Int. Cl.**
G06F 21/00 (2013.01)
G06F 21/53 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***G06F 21/53*** (2013.01); ***G06F 21/51*** (2013.01); ***G06F 21/56*** (2013.01); ***H04L 63/145*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC G06F 21/52; G06F 21/53; G06F 21/56; G06F 17/30144; H04L 67/06; H04L 63/145
USPC 726/1
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,562,305 A 12/1985 Gaffney, Jr.
4,864,616 A 9/1989 Pond et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0636977 7/1994
EP 0813133 12/1997
(Continued)

OTHER PUBLICATIONS

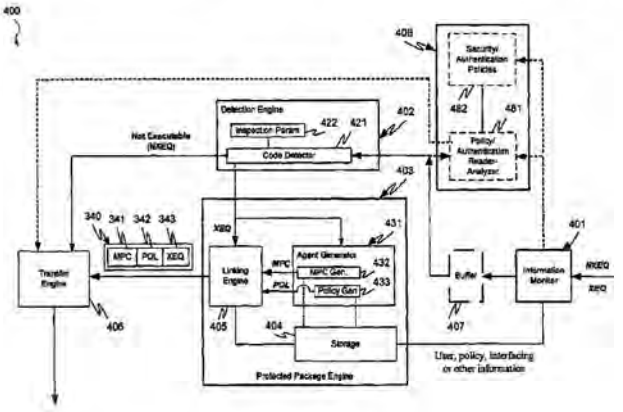
Leroy, Xavier, and Francois Rouaix. "Security properties of typed applets." Secure Internet Programming. Springer Berlin Heidelberg. 1999. 147-182.*
(Continued)

Primary Examiner — Syed Zaidi
(74) *Attorney, Agent, or Firm* — Dawn-Marie Bey; Bey & Cotropia PLLC

(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 TN 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 for monitoring information received, 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. 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.

16 Claims, 10 Drawing Sheets



Related U.S. Application Data

12/471,942, filed on May 26, 2009, now Pat. No. 8,079,086, which is a continuation of application No. 11/370,114, filed on Mar. 7, 2006, now Pat. No. 7,613,926, which is a continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, which is 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. 09/861,229 is 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.

(60) Provisional application No. 60/205,591, filed on May 17, 2000, provisional application No. 60/030,639, filed on Nov. 8, 1996.

(51) **Int. Cl.**

G06F 21/51	(2013.01)
H04L 29/06	(2006.01)
G06F 21/56	(2013.01)

(52) **U.S. Cl.**

CPC ***H04L 63/1441*** (2013.01); ***H04L 63/168*** (2013.01); ***H04L 63/20*** (2013.01); *G06F 2221/033* (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,978,484	A	12/1990	Takashige et al.	264/40.1
5,050,212	A	9/1991	Dyson	380/25
5,077,677	A	12/1991	Murphy et al.	
5,263,147	A	11/1993	Francisco et al.	
5,278,901	A	1/1994	Shieh et al.	
5,283,830	A	2/1994	Hinsley et al.	
5,311,591	A	5/1994	Fischer	
5,313,616	A	5/1994	Cline et al.	395/500
5,319,776	A	6/1994	Rile et al.	
5,337,360	A	8/1994	Fischer	
5,345,595	A	9/1994	Johnson et al.	
5,359,659	A	10/1994	Rosenthal	
5,361,359	A	11/1994	Tajalli et al.	
5,389,196	A	2/1995	Bloomstein et al.	
5,398,196	A	3/1995	Chambers	
5,412,717	A	5/1995	Fischer	
5,414,833	A	5/1995	Hershey et al.	
5,421,006	A	5/1995	Jablon et al.	395/575
5,440,723	A	8/1995	Arnold et al.	
5,452,442	A	9/1995	Kephart	
5,471,614	A	11/1995	Kakimoto	
5,475,753	A	12/1995	Barbara et al.	
5,483,649	A	1/1996	Kuznetsov et al.	
5,485,409	A	1/1996	Gupta et al.	
5,485,575	A	1/1996	Chess et al.	
5,524,238	A	6/1996	Miller et al.	
5,572,643	A	11/1996	Judson	
5,579,509	A	11/1996	Furtney et al.	
5,606,668	A	2/1997	Shwed	
5,621,889	A	4/1997	Lermuzeaux et al.	
5,623,600	A	4/1997	Ji et al.	
5,623,601	A	4/1997	Vu	
5,625,693	A	4/1997	Rohatgi et al.	380/23
5,638,446	A	6/1997	Rubin	
5,666,411	A	9/1997	McCarty	
5,673,316	A	9/1997	Auerbach et al.	
5,675,711	A	10/1997	Kephart et al.	
5,678,041	A	10/1997	Baker et al.	
5,680,461	A	10/1997	McManis	
5,692,047	A	11/1997	McManis	
5,692,124	A	11/1997	Holden et al.	
5,696,822	A	12/1997	Nachenberg	
5,699,512	A	12/1997	Moriyasu et al.	
5,720,033	A	2/1998	Ceo	
5,724,425	A	3/1998	Chang et al.	
5,740,248	A	4/1998	Fieres et al.	
5,740,441	A	4/1998	Yellin et al.	
5,748,960	A	5/1998	Fischer	395/683
5,757,915	A	5/1998	Aucsmith et al.	
5,761,421	A	6/1998	Van Hoff et al.	
5,765,030	A	6/1998	Nachenberg et al.	
5,765,205	A	6/1998	Breslau et al.	
5,784,459	A	7/1998	Devarakonda et al.	
5,787,175	A	7/1998	Carter	
5,796,952	A	8/1998	Davis et al.	
5,805,829	A	9/1998	Cohen et al.	
5,809,230	A	9/1998	Pereira	
5,815,709	A	9/1998	Waldo et al.	
5,825,877	A	10/1998	Dan et al.	
5,832,208	A	11/1998	Chen et al.	
5,832,274	A	11/1998	Cutler et al.	
5,842,002	A	11/1998	Schnurer et al.	
5,842,040	A	11/1998	Hughes et al.	
5,845,281	A	12/1998	Benson et al.	
5,850,559	A	12/1998	Angelo et al.	
5,854,916	A	12/1998	Nachenberg	
5,859,966	A	1/1999	Hayman et al.	
5,860,011	A	1/1999	Kolawa et al.	
5,864,683	A	1/1999	Boebert et al.	
5,867,651	A	2/1999	Dan et al.	
5,878,258	A	3/1999	Pizi et al.	
5,881,151	A	3/1999	Yamamoto	
5,884,003	A	3/1999	Cloud et al.	385/135
5,884,033	A	3/1999	Duvall	
5,889,943	A	3/1999	Ji et al.	
5,889,952	A	3/1999	Hunnicutt et al.	
5,892,904	A	4/1999	Atkinson et al.	
5,894,516	A	4/1999	Brandenburg	
5,911,043	A	6/1999	Duffy et al.	
5,933,503	A	8/1999	Schell et al.	
5,940,590	A	8/1999	Lynne et al.	
5,944,821	A	8/1999	Angelo	713/200
5,949,415	A	9/1999	Lin et al.	
5,951,698	A	9/1999	Chen et al.	
5,956,481	A	9/1999	Walsh et al.	
5,958,050	A *	9/1999	Griffin et al.	726/1
5,960,170	A *	9/1999	Chen et al.	714/38.14
5,963,742	A	10/1999	Williams	
5,964,889	A	10/1999	Nachenberg	
5,974,549	A	10/1999	Golan	
5,978,484	A	11/1999	Apperson et al.	
5,978,840	A	11/1999	Nguyen et al.	
5,983,267	A	11/1999	Shklar et al.	
5,983,348	A *	11/1999	Ji	726/13
5,987,123	A	11/1999	Scott et al.	
5,987,610	A	11/1999	Franczek et al.	
5,987,611	A	11/1999	Freund	
5,991,810	A	11/1999	Shapiro et al.	
6,021,510	A	2/2000	Nachenberg	
6,049,671	A	4/2000	Slivka et al.	395/712
6,058,482	A	5/2000	Liu	
6,065,118	A	5/2000	Bull et al.	
6,067,575	A	5/2000	McManis et al.	
6,070,239	A	5/2000	McManis	
6,073,175	A	6/2000	Tavs et al.	
6,083,279	A	7/2000	Cuomo et al.	
6,088,801	A	7/2000	Grecsek	
6,088,803	A	7/2000	Tso et al.	
6,092,147	A	7/2000	Levy et al.	
6,092,194	A	7/2000	Touboul	
6,125,390	A	9/2000	Touboul	
6,128,774	A	10/2000	Necula et al.	
6,148,083	A	11/2000	Fieres et al.	380/255
6,148,401	A	11/2000	Devanbu et al.	
6,154,842	A	11/2000	Van Peursem	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
6,167,522	A	12/2000	Lee et al.	
6,170,012	B1	1/2001	Coss et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,170,057 B1 1/2001 Inoue et al.
6,175,924 B1 1/2001 Arnold
6,192,512 B1 2/2001 Chess
6,205,551 B1 3/2001 Grosse
6,230,288 B1 5/2001 Kuo et al.
6,253,323 B1 6/2001 Cox et al.
6,253,370 B1 6/2001 Abadi et al.
6,263,442 B1 7/2001 Mueller et al.
6,266,664 B1 7/2001 Russell-Falla et al.
6,266,773 B1 7/2001 Kisor et al.
6,272,641 B1 8/2001 Ji
6,275,937 B1 8/2001 Hailpern et al.
6,275,938 B1 * 8/2001 Bond et al. 726/23
6,317,868 B1 11/2001 Grimm et al.
6,321,334 B1 11/2001 Jerger et al.
6,324,685 B1 11/2001 Balassanian
6,336,140 B1 1/2002 Elgressy et al.
6,339,829 B1 1/2002 Beadle et al.
6,343,376 B1 1/2002 Saxe et al.
6,351,816 B1 2/2002 Mueller et al.
6,357,008 B1 3/2002 Nachenberg
6,367,012 B1 4/2002 Atkinson et al.
6,401,241 B1 6/2002 Taylor
6,418,444 B1 7/2002 Raduchel et al.
6,425,058 B1 7/2002 Arimilli et al.
6,434,668 B1 8/2002 Arimilli et al.
6,434,669 B1 8/2002 Arimilli et al.
6,449,723 B1 9/2002 Elgressy et al.
6,470,450 B1 10/2002 Langford et al.
6,473,800 B1 10/2002 Jerger et al.
6,480,962 B1 11/2002 Touboul
6,487,666 B1 11/2002 Shanklin et al.
6,505,300 B2 1/2003 Chan et al.
6,519,679 B2 2/2003 Devireddy et al.
6,526,565 B1 2/2003 Nally
6,560,774 B1 5/2003 Gordon et al.
6,567,917 B1 5/2003 Ziese
6,571,338 B1 5/2003 Shaio et al.
6,594,783 B1 7/2003 Dollin et al.
6,598,033 B2 7/2003 Ross et al.
6,604,143 B1 8/2003 Nagar et al.
6,622,247 B1 9/2003 Isaak
6,622,781 B2 9/2003 Chou
6,640,284 B1 10/2003 Shaw et al.
6,643,696 B2 11/2003 Davis et al.
6,668,325 B1 12/2003 Collberg et al.
6,691,230 B1 * 2/2004 Bardon 726/27
6,697,950 B1 2/2004 Ko
6,704,798 B1 3/2004 Mogul
6,721,721 B1 4/2004 Bates et al.
6,725,424 B1 4/2004 Schwerdtfeger et al.
6,732,179 B1 5/2004 Brown et al.
6,748,538 B1 6/2004 Chan et al.
6,772,214 B1 8/2004 McClain et al.
6,772,225 B1 8/2004 Jennings, III et al.
6,772,332 B1 8/2004 Boebert et al.
6,775,536 B1 8/2004 Geiger et al. 455/411
6,804,780 B1 10/2004 Touboul
6,829,746 B1 12/2004 Schwerdtfeger et al.
6,832,222 B1 12/2004 Zimowski
6,842,861 B1 1/2005 Cox et al.
6,886,013 B1 4/2005 Beranek
6,901,519 B1 5/2005 Stewart et al.
6,917,953 B2 7/2005 Simon et al.
6,917,960 B1 7/2005 Decasper et al.
6,944,772 B2 9/2005 Dozortsev
6,981,138 B2 12/2005 Douceur et al.
6,996,841 B2 2/2006 Kadyk et al.
6,996,845 B1 2/2006 Hurst et al. 726/25
7,039,946 B1 5/2006 Binding
7,047,369 B1 5/2006 Gruper et al.
7,051,164 B2 5/2006 Smith
7,052,822 B2 5/2006 Goto 430/283.1
7,058,822 B2 6/2006 Edery et al.
7,076,650 B1 7/2006 Sonnenberg
7,089,591 B1 8/2006 Chi
7,093,135 B1 8/2006 Radatti et al.
7,096,496 B1 8/2006 Challener et al.
7,103,714 B1 9/2006 Jacobs et al.
7,143,444 B2 11/2006 Porras et al.
7,181,725 B1 2/2007 Posegga et al.
7,185,210 B1 2/2007 Faden 713/200
7,210,041 B1 4/2007 Gryaznov et al.
7,243,366 B2 7/2007 Medvinsky et al.
7,308,648 B1 12/2007 Buchthal et al.
7,321,970 B2 1/2008 Watkins et al.
7,343,604 B2 3/2008 Grabamik et al.
7,350,204 B2 3/2008 Lambert et al.
7,363,361 B2 4/2008 Tewari et al.
7,363,493 B2 4/2008 Dotan
7,398,553 B1 7/2008 Li
7,409,718 B1 8/2008 Hong et al. 726/24
7,418,731 B2 8/2008 Touboul
7,418,732 B2 8/2008 Campbell et al.
7,434,260 B2 10/2008 Hong et al.
7,461,262 B1 12/2008 O'Toole, Jr.
7,467,304 B2 12/2008 Bar-El et al.
7,487,540 B2 2/2009 Shipp
7,493,658 B2 2/2009 Shipp
7,506,155 B1 3/2009 Stewart et al.
7,567,800 B2 7/2009 Uematsu et al.
7,568,114 B1 7/2009 Schlafly
7,574,738 B2 8/2009 Daude et al.
7,587,755 B2 9/2009 Kramer 726/4
7,613,918 B2 11/2009 Ben-Itzhak 713/150
7,613,926 B2 11/2009 Edery et al.
7,620,733 B1 11/2009 Tzakikario et al.
7,627,896 B2 12/2009 Herrmann
7,636,851 B2 12/2009 Hamblin et al.
7,636,945 B2 12/2009 Chandnani et al.
7,640,427 B2 12/2009 Callas et al.
7,647,633 B2 1/2010 Edery et al.
7,660,902 B2 2/2010 Graham et al.
7,694,135 B2 4/2010 Rowan et al.
7,694,323 B2 4/2010 LeJoie et al.
7,694,328 B2 4/2010 Joshi et al.
7,698,549 B2 4/2010 Thornton et al.
7,735,138 B2 6/2010 Zhao
7,752,242 B1 7/2010 Satish et al. 707/826
7,756,996 B2 7/2010 Touboul
7,757,289 B2 7/2010 Gruzman et al.
7,761,594 B1 7/2010 Mowat
7,761,605 B1 7/2010 Rothwell et al.
7,774,619 B2 8/2010 Paaske et al.
7,797,727 B1 9/2010 Miller et al. 726/1
7,805,758 B2 9/2010 Hasegawa et al.
7,822,974 B2 10/2010 Kinser et al.
7,849,306 B2 12/2010 Takeshima et al.
7,865,961 B2 1/2011 Hasegawa et al.
7,930,299 B2 4/2011 Yuval et al.
7,930,738 B1 4/2011 Petersen 726/16
7,975,305 B2 7/2011 Rubin et al.
7,984,304 B1 7/2011 Waldspurger et al.
7,984,479 B2 7/2011 Brabson et al.
8,015,182 B2 9/2011 Yuval et al.
8,078,740 B2 12/2011 Franco et al.
8,079,086 B1 12/2011 Edery et al.
8,126,866 B1 2/2012 Barton et al.
8,141,154 B2 3/2012 Gruman et al. 726/22
8,225,392 B2 7/2012 Dubrovsky
8,225,408 B2 7/2012 Rubin et al.
8,244,910 B2 8/2012 Davis
8,281,401 B2 10/2012 Pennington et al.
8,321,662 B2 11/2012 Hazlewood et al.
8,347,392 B2 1/2013 Chess et al.
8,364,951 B2 1/2013 Peterka et al.
8,374,958 B2 2/2013 Blott et al.
8,375,423 B2 2/2013 Hays et al. 726/5
8,402,012 B1 3/2013 Herz et al.
8,424,090 B2 4/2013 Kang et al. 726/24
8,438,499 B2 5/2013 Dixon et al.
8,522,350 B2 8/2013 Davenport et al. 726/24
8,677,494 B2 3/2014 Edery et al.
2001/0025311 A1 9/2001 Arai et al.

(56) **References Cited**

U.S. PATENT DOCUMENTS

2002/0026511	A1	2/2002	Garcia-Luna-Aceves et al.	
2002/0059144	A1	5/2002	Meffert et al.	
2002/0059157	A1	5/2002	Spooner et al.	
2002/0065775	A1	5/2002	Monaghan	
2002/0065776	A1	5/2002	Calder et al.	
2002/0066024	A1	5/2002	Schmall et al.	
2002/0073330	A1	6/2002	Chandnani et al.	
2002/0099829	A1	7/2002	Richards et al.	
2002/0165912	A1	11/2002	Wenocur et al.	
2002/0198953	A1	12/2002	O'Rourke et al.	
2003/0009571	A1	1/2003	Bavadekar	
2003/0014662	A1	1/2003	Gupta et al.	
2003/0028616	A1	2/2003	Aoki et al.	
2003/0046412	A1	3/2003	Tsunoda et al.	
2003/0063752	A1	4/2003	Medvinsky et al.	
2003/0074190	A1	4/2003	Allison	
2003/0097582	A1	5/2003	Audebert et al.	
2003/0097591	A1	5/2003	Pham et al.	
2003/0101358	A1	5/2003	Porras et al.	
2003/0154265	A1	8/2003	Raffaele et al.	
2003/0172291	A1	9/2003	Judge et al.	
2003/0191801	A1	10/2003	Paul	
2003/0208490	A1	11/2003	Larrea et al.	
2003/0225883	A1	12/2003	Greaves et al.	
2004/0006706	A1	1/2004	Erlingsson	713/200
2004/0015538	A1	1/2004	Agarwalla et al.	
2004/0015723	A1	1/2004	Pham et al.	
2004/0016948	A1	1/2004	Lin	
2004/0039924	A1	2/2004	Baldwin et al.	
2004/0054854	A1	3/2004	Thiyagaranjan et al.	
2004/0073668	A1	4/2004	Bhat et al.	
2004/0073811	A1	4/2004	Sanin	
2004/0088425	A1	5/2004	Rubinstein et al.	
2004/0107363	A1	6/2004	Monteverde	
2004/0111413	A1	6/2004	Birbo et al.	
2004/0117426	A1	6/2004	Rudkin et al.	
2004/0117623	A1	6/2004	Kalogridis et al.	
2004/0122926	A1	6/2004	Moore et al.	
2004/0133520	A1	7/2004	Callas et al.	
2004/0148281	A1	7/2004	Bates et al.	
2004/0158741	A1	8/2004	Schneider	
2004/0230744	A1	11/2004	Teh	
2004/0243839	A1	12/2004	Bhatia et al.	
2004/0249965	A1	12/2004	Huggins et al.	
2005/0021791	A1	1/2005	Sakiyama et al.	709/229
2005/0050338	A1	3/2005	Liang et al.	
2005/0114653	A1	5/2005	Sudia	
2005/0172338	A1	8/2005	Sandu et al.	
2005/0193429	A1	9/2005	Demopoulos et al.	
2005/0246383	A1	11/2005	Desai et al.	
2006/0031207	A1	2/2006	Bjamestam et al.	
2006/0031352	A1	2/2006	Marston et al.	
2006/0048224	A1	3/2006	Duncan et al.	
2006/0143442	A1	6/2006	Smith	
2006/0236392	A1	10/2006	Thomas et al.	
2006/0294366	A1	12/2006	Nadalin et al.	
2007/0011739	A1	1/2007	Zamir et al.	
2007/0016770	A1	1/2007	Dumitru	
2007/0113282	A1	5/2007	Ross	
2007/0134082	A1	6/2007	Blomgren et al.	
2008/0066160	A1	3/2008	Becker et al.	
2008/0083012	A1	4/2008	Yu et al.	726/1
2009/0125612	A1	5/2009	Rabetge et al.	
2009/0193497	A1	7/2009	Kikuchi et al.	726/1
2010/0195909	A1	8/2010	Wasson et al.	
2011/0099621	A1	4/2011	Lizarraga et al.	
2012/0117651	A1	5/2012	Edery et al.	

FOREIGN PATENT DOCUMENTS

EP	0909072		4/1999
EP	0909075		4/1999
EP	1021276		7/2000
EP	1 058 199	A2	12/2000
EP	1091276		4/2001
EP	1119151		7/2001
EP	1132796		9/2001
IL	120420		12/1999
IL	129729		5/2005
JP	08-263447		10/1996
JP	H 1021134		1/1998
JP	01-256045	A	9/2001
JP	2002183019		6/2002
JP	02-358229		12/2002
JP	02-358253	A	12/2002
JP	2002373108		12/2002
JP	2003177991		6/2003
JP	3952315		5/2007
WO	WO 93/25024		12/1993
WO	WO 95/27249		10/1995
WO	WO95/33237		12/1995
WO	WO 97/12322		4/1997
WO	WO 97/31698		9/1997
WO	WO 98/21683		5/1998
WO	WO 98/31124		7/1998
WO	WO 98/40993		9/1998
WO	WO 99/16225		4/1999
WO	WO 99/35583		7/1999
WO	WO 00/08793		2/2000
WO	WO 01/22682	A2	3/2001
WO	WO 01/25882		4/2001
WO	WO 01/88673		11/2001
WO	WO 01/98907		12/2001
WO	WO 01/99374		12/2001
WO	WO 02/13479		2/2002
WO	WO 2004/063948		7/2004
WO	WO 2006/025050		3/2006

OTHER PUBLICATIONS

Donald E. Knuth, "The Art of Computer Programming," vol. 3 (1st Ed., 1973).

Donald E. Knuth, "The Art of Computer Programming," vol. 3, 2nd Ed., 1998).

"100 or So Books that Shaped a Century of Science," American Scientist, vol. 87, No. 6, pp. 542-544, 546, 549-550, 553 (Nov.-Dec. 1999).

Obituary of Frances E. Holberton, 84, "Early Computer Programmer," published Dec. 17, 2001.

Decision, dated Nov. 19, 2013, granting ex parte reexamination for U.S. Appl. No. 90/013,016, 17 pp.

Non-Final Office Action, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,016, 14 pp.

Decision, dated Dec. 6, 2013, granting ex parte reexamination for U.S. Appl. No. 90/013,017, 20 pp.

Non-Final Office Action, dated Dec. 6, 2013, for U.S. Appl. No. 90/013,017, 42 pp.

Final Office Action, dated Sep. 8, 2014, for U.S. Appl. No. 90/013,017, 51 pp.

Advisory Action, dated Oct. 23, 2014, for U.S. Appl. No. 90/013,017, 12 pp.

Examiner's Answer, dated Mar. 30, 2015, for U.S. Appl. No. 90/013,017, 117 pp.

Decision, dated Jan. 25, 2012, granting inter partes reexamination for U.S. Appl. No. 95/001,836, 77 pp.

Non-Final Office Action, dated Jan. 25, 2012, for U.S. Appl. No. 95/001,836, 17 pp.

Action Closing Prosecution dated Oct. 2, 2013, for U.S. Appl. No. 95/001,836, 61 pp.

Right of Appeal Notice, dated Jun. 27, 2014, for U.S. Appl. No. 95/001,836, 82 pp.

Examiner's Answer, dated Mar. 25, 2015, for U.S. Appl. No. 95/001,836, 4 pp.

Order Denying Request for Ex Parte Reexamination, dated Mar. 27, 2014, for U.S. Appl. No. 90/013,147, 26 pp.

Order Granting Request for Inter Partes Reexamination, dated Aug. 28, 2007, for U.S. Appl. No. 90/008,678, 13 pp.

Non-Final Office Action, dated May 8, 2008, for U.S. Appl. No. 90/008,678, 18 pp.

(56) **References Cited**

OTHER PUBLICATIONS

Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Feb. 27, 2009, 11 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,480,962, dated May 26, 2009, U.S. Appl. No. 90/008,678, 2 pp.
Order Granting Request for Ex Parte Reexamination, dated Aug. 28, 2007, for U.S. Appl. No. 90/008,684, 11 pp.
Non-Final Office Action, dated Dec. 19, 2008, for U.S. Appl. No. 90/008,684, 39 pp.
Final Office Action, dated Nov. 20, 2009, for U.S. Appl. No. 90/008,684, 53 pp.
Advisory Action, dated Feb. 3, 2010, for U.S. Appl. No. 90/008,684, 4 pp.
Examiner's Answer, dated Sep. 10, 2010, for U.S. Appl. No. 90/008,684, 86 pp.
Decision on Appeal, Examiner Affirmed, dated Jun. 6, 2011, for U.S. Appl. No. 90/008,684, 8 pp.
United States Patent and Trademark Office's Response to Appellant's Brief, dated Feb. 13, 2012, for U.S. Appl. No. 90/008,684.
Judgment of United States Court of Appeal for the Federal Circuit, dated Jan. 11, 2013, for U.S. Appl. No. 90/008,684, 2 pp.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Jul. 26, 2013, for U.S. Appl. No. 90/008,684, 6 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,167,520, dated Sep. 6, 2013, U.S. Appl. No. 90/008,684, 2 pp.
Order Granting Request for Ex Parte Reexamination (U.S. Pat. No. 6,092,194), dated Aug. 28, 2008, for U.S. Appl. No. 90/009,175, 11 pp.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated May 28, 2009, for U.S. Appl. No. 90/009,175, 21 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,092,194, dated Aug. 18, 2009, U.S. Appl. No. 90/009,175, 15 pp.
"Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns," by Morton Swimmer, in Proceedings of the 1995 Virus Bulletin Conference.
"Proof-Carrying Code," George C. Necula and Peter Lee. Technical Report CMU-CS-96-165, Nov. 1996.
"A Flexible Security Model for Using Internet Content," by Nayeem Islam et al., in Proceedings of the Sixteenth Symposium on Reliable Distributed Systems (1997).
"MIME Encapsulation of Aggregate Applet Objects" by Alireza Bahreman et al., IETF, Jun. 1996.
"An Immune System for Cyberspace," by Jeffrey O. Kephart et al., in Proc. of the IEEE SMC'97.
"A Generic Virus Detection Agent on the Internet," by Jieh-Sheng Lee et al., in the Proceedings of the Thirtieth Hawaii International Conference on System Sciences 1997.
"Understanding Heuristics: Symantec's Bloodhound Technology," Symantec White Paper Series vol. XXXIV (1997).
"Understanding Symantec's Anti-virus Strategy for Internet Gateways," Symantec Enterprise Papers vol. XXX (1997).
"Understanding and Managing Polymorphic Viruses," Symantec Enterprise Papers vol. XXX (1997).
"Symantec Announces Norton Antivirus for Firewalls for Maximum Protection From Internet-Borne Viruses," Symantec Press Release (1997) [retrieved on Feb. 23, 2015], located at http://www.symantec.com/about/news/release/article.jsp?prid=19970506_01.
Bruce Schneier, Applied Cryptography, John Wiley & Sons Inc. (1994).
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance, Proceedings of the USENIX Security Symposium, San Francisco, CA (Aug. 5-9, 2002).
Nick Feamster et al., Thwarting Web Censorship with Untrusted Messenger Discovery, 3rd Workshop on Privacy Enhancing Technologies, Dresden, Germany, Mar. 2003.
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance (Presentation), 11th Usenix Security Symposium, Aug. 8, 2002.
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance (Presentation), Harvard Kennedy School, Nov. 27, 2002.
ePolicy Orchestrator Version 2.0 Product Guide (2001).
Common Updater Version 1.0 Configuration Guide (2002).
ePolicy Orchestrator Version 3.0 Product Guide (2003).
ePolicy Orchestrator Version 3.0 Installation Guide (2003).
William G.J. Halfond & Alessandro Orso, Combining Static Analysis & Runtime Monitoring to Counter SQL-Injection Attacks, Workshop on Dynamic Analysis, ACM, May 17, 2005.
Yougang Song & Brett D. Fleisch, "Rico: a security proxy for mobile code," Elsevier, Jun. 2004.
James Newsome and Dawn Song, Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation, May 2004.
D.P. Ghormley et al., SLIC: An Extensibility System for Commodity Operating Systems, USENIX Annual Technical Conference, Jun. 1998.
George C. Necula, "Proof-carrying code." in Proceedings of the 24th ACM Symposium on Principles of Programming Languages, Paris, France, Jan. 1997.
Encapsulating Mobile Objects, by Hermann Hartig et al., in the Proceedings of the 17th International Conference on Distributed Computing Systems, May, 1997.
Christopher Small, "A Tool for Constructing Safe Extensible C++ Systems," in Proceedings of the Third USENIX Conference on Object-Oriented Technologies (1997).
"Symantec's Norton AntiVirus 4.0 Delivers Multi-Platform Support," Symantec (1997) (Sep. 23, 1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970923_01.
"Symantec's AntiVirus Research Center Responds to Mutating Macro Viruses With Bloodhound-Macro Technology," Symantec (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970923_03.
"Symantec Announces Norton Internet Email Gateway at Internet World," Symantec Press Release (1996) http://www.symantec.com/about/news/release/article.jsp?prid=19961211_03.
"Symantec Ships Norton AntiVirus for Internet Email Gateways," Symantec Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970224_01.
"Symantec Announces Norton Antivirus for Firewalls for Maximum Protection From Internet-Borne Viruses," Symantec Press Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970506_01.
"Symantec, the Worldwide Leader in Antivirus, Ships Norton Antivirus for Firewalls," Symantec Press Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970714_01.
Check Point, "Check Point Software Delivers Breakthrough Security Advancements with FireWall-1.3.0," Oct. 7, 1996, http://www.checkpoint.com/press/1996/30release.html.
Virus Bulletin, International Conference, Sep. 20-22, 1995.
U.S. Appl. No. 60/174,952, filed Jan. 7, 2000, Lieno, et al.
U.S. Appl. No. 60/163,361, filed Nov. 3, 1999, Geiger, et al.
U.S. Appl. No. 60/156,872, 9/39/99, Huang, et al.
U.S. Appl. No. 60/104,072, filed Oct. 13, 1998, Hicken.
U.S. Appl. No. 60/093,559, filed Jul. 21, 1998, Maloney, et al.
Mary R. Thompson, et al., "Certificate-Based Authorization Policy in a PKI Environment," Nov. 2003.
P. C. Cheng, "An Architecture for the Internet Key Exchange Protocol," 2001.
List of Secure Internet Programming Publications, [retrieved on Dec. 22, 2010], 4 pp., Retrieved from the Internet: www.cs.princeton.edu/sip/pub/index.php3.
"Synthesizing Fast Intrusion Prevention/Detection Systems From High-Level Specifications," by Sekar, et al. (1999).
"A Web Navigator With Applets in Caml," by Francois Ronaix, Fifth International World Wide Web Conference, Paris, France, May 6-10, 1996, 20 pp.
Slade, Robert, "Guide to Computer Viruses: How to Avoid Them, How to Get Rid of Them, and How to Get Help" (Apr. 1996).
Garfinkel, Simson and Spafford, Gene, "Practical Unix and Internet Security," O'Reilly & Associates, Inc., 1996.
ThunderBYTE Anti-Virus Utilities User Manual, ThunderBYTE B.V., Wijchen, The Netherlands, 1996.

(56) **References Cited**

OTHER PUBLICATIONS

"MCF: A Malicious Code Filter," by Lo, et al., 27 pp. (May 4, 1994).
"Towards a Testbed for Malicious Code Detection," by Lo, et al., Compcon Spring '91 Digest of Papers at 160-166, IEEE, 1991.
"Frequently Asked Questions—Java Security," Copyright 1995-1998 [retrieved on Dec. 15, 2011], 13 pp., Retrieved from the Internet: mhtml:file://S:\DFSR_DATA\Data03\030052\01_Finjan_v._Symantec(DDEL-00593)\Prio . . .
Marcionek, David, "A Complete ActiveX Web Control Tutorial," Jun. 21, 2006 [retrieved on Sep. 29, 2014], 22 pp., Retrieved from the Internet: http://www.codeproject.com/Articles/14533/A-Complete-ActiveX-Web-Control-Tutorial.
Greenberg, Ross M., "Flu_Shot+, Version 1.81, A Form of Protection From Viral and Trojan Programs," Copyright 1988-1990, 54 pp.
"NetShield Version 1.61 Release Candidate for Novell NetWare v4.01 and Novell NetWare for OS/2 v4.01," McAfee Associates, Inc., Copyright 1994, 54 pp.
Balfanz, et al., "A Java Filter," 1997.
Ranum, et al., "A Toolkit and Methods for Internet Firewalls," 1994.
Pope, Nick, "Standards in Commercial Security," Designing Secure Systems, IEE Colloquium, Jun. 19, 1992.
Whyman, T., "X.400 Security Features," Security and Networks, IEE Colloquium, Jun. 25, 1990.
Norton AntiVirus™ for Windows® 95 User's Guide, Published by Symantec in 1995 (179 pp.).
Jaeger, et al., "Building Systems that Flexibly Control Downloadable Executable Content," Proceedings of the Sixth USENIX UNIX Security Symposium, Jul. 1996 (19 pp.).
Rasmusson, Andreas and Jansson, Sverker,"Personal Security Assistance for Secure Internet Commerce," Sep. 16, 1996 (12 pp.), Available as dvi, ps, html, at http://www.sics.se/~ara/papers/NSP96.html.
Bharat, et al., "Migratory Applications," Nov. 15, 1995 (10 pp.).
Dean, Drew, et al., "Java Security: From HotJava to Netscape and Beyond," 1996 IEEE Symposium on Security and Privacy, May 6, 1996 (11 pp.).
Sterbenz, Andreas, "An Evaluation of the Java Security Model," IEEE, Dec. 1996 (13 pp.).
Fritzinger, J. Steven, et al., Java Security, Sun Microsystems, Dec. 1996 (7 pp.).
Bank, Joseph A., "Java Security," Dec. 8, 1995 (14 pp.).
Chappell, "Understanding ActiveX and OLE: A Guide for Developers and Managers (Strategic Technology)," Sep. 1, 1996, Microsoft Press (91 pp.).
Crosbie, et al., "Active Defense of a Computer System Using Autonomous Agents," Feb. 15, 1995 (14 pp.).
"Trend Micro's Virus Protection Added to Sun Microsystems Netra Internet Servers," Business Wire, Oct. 1, 1996, available at http://www.cs.indiana.edu/~kinzler/pubs/viruswall.html.
"Symantec Announces Norton Antivirus 2.0 for Windows NT," Symantec Corporation press release, Sep. 16, 1996, available at http://www.symantec.com/about/news/release/article.jsp?prid=19960916_01.
"Dark Avenger Mutation Engine No Threat to Protected PCs," McAfee, Inc., press release, May 11, 1992, available at http://securitydigest.org/virus/mirror/www.phreak.orgvirus1/1992/vin105.191.
Gryaznov, D. O., "Scanners of the Year 2000: Heuristics," Proceedings of the Fifth International Virus Bulletin Conference, pp. 225-234 (1995), available at http://vxheavens.com/lib/adg00.html.
"Symantec Announces Norton Internet Email Gateway at Internet World—Booth # 369 on Dec. 11, 12, and 13," Symantec Corporation press release, Dec. 11, 1996, available at http://www.symantec.com/about/news/release/article.jsp?prid=19961211_03.
"Internet Security Gets Less Costly and Easier to Manage: Integralis Announces MIMEsweeper Compatible with Check Point FireWall-1 on Single NT Server; E-mail virus detection and content management can reside on Firewall server, saving money and supports costs" [online], The Free Library, Sep. 16, 1996 [retrieved on Mar. 20, 2012], 4 pp., Retrieved From the Internet: http://www.thefreelibrary.com/Internet+Security+gets+Less+Costly+and+Easier+to+Manage%3a+Integralis . . . a018675791%22%3Einternet%20Security%20gets%20Less%20Costly%20and%20Easier%20to%20Manage:%20Integralis%20Announces%20MIMEsweeper%20Compatible%20with%20Check%20Point%20FireWall-1%20on%20Single%20NT%20Server;%20E-mail%20virus%20detection%20and%20content%20management%20can%20reside%20on%20Firewall%20server,%20saving%20money%20and%20support%20costs (Current Link: http://www.thefreelibrary.com/Internet Security gets Less Costly and Easier to Manage: Integralis . . . -a018675791).
"The Java Language Specification," by Gosling, et al. (1996).
"The Java Virtual Machine Specification," by Tim Lindholm and Frank Yellin (1997).
"Java Security and a Firewall Extension for Authenticity Control of Java Applets," by Magnus Johansson (Jan. 29, 1997).
"Static Analysis of Programs With Application to Malicious Code Detection," by Raymond Lo, Dissertation, University of California, 193 pp. (1992).
File History for U.S. Pat. No. 6,804,780.
"Virus Detection Alternatives," by Patrick Min, http://vx.netlux.org/lib/apm00.html (Jul. 1992).
"Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns," by LeCharlier, et al. (Sep. 1995).
The Giant Black Book of Computer Viruses by Mark Ludwig (1995).
HotJava: The Security Story.
"The Java Filter," [retrieved on Jan. 2, 2011], 1 p., Retrieved from the Internet: http://www.cs.princeton.edu/sip/JavaFilter/index.html.
"Improved JavaScript and Java Screening Function," by Claunch (May 4, 1996), available at http://groups.google.com/group/muc.lists.firewalls/msg/96bc8043476ab0b7.
"New Version of Java, JavaScript, ActiveX Screening," by Claunch (Jul. 3, 1996), available at http://groups.google.com/group/muc.lists.firewalls/msg/41a45c2a9d5373f2.
"Identifying and Controlling Undesirable Program Behaviors," by Maria King, Proceedings of the 14th National Computer Security Conference, 1991.
"PACL's: An Access Control List Approach to Anti-Viral Security," by Wichers, et al., in USENIX Workshop Proceedings, UNIX Security II, pp. 71-82, 1990.
Endrijonas, Janet, "Rx PC the Anti-Virus Handbook," Published in the U.S. in 1993 by Tab Books, a division of McGraw-Hill, Inc. (201 pp.).
"Secure Code Distribution," by X. Nick Zhang (Jun. 1997).
Omura, Jim K., "Novel Applications of Cryptography in Digital Communications," IEEE Communications Magazine, pp. 21-29, May 1990.
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/cgibin/iel_cgi?se . . . 2ehts%26ViewTemplate%3ddocview%5fb%2ehts.
IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, pp. 6-7, Nov. 15, 1995.
Leach, Norvir, et al., "IE 3.0 Applets Will Earn Certification," PC Week, vol. 13,No. 29, 2 pp., Jul. 22, 1996.
"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.
"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.
Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet," Microsoft Corporation, including Abstract, Contents, Introduction, and pp. 1-10, Oct. 1996.
"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.
Company Profile, "Finjan-Safe Surfing. The Java Security Solutions Provider," Article published on the Internet by Finjan Software Ltd., 3 pp., Oct. 31, 1996.
"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0," Las Vegas Convention Center/Pavillion 5 P5551, 3 pp., Nov. 18, 1996.

(56) **References Cited**

OTHER PUBLICATIONS

"Java Security: Issues & Solutions," Article published on the Internet by Finjan Software Ltd., 10 pp., 1996.
"Products," Article published on the Internet, "Revolutionary Security for a New Computing Paradigm" regarding SurfinGate™, 7 pp.
Mark LaDue, "Online Business Consultant: Java Security: Whose Business Is It?," Article published on the Internet, Home Page Press, Inc., 4 pp., 1996.
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/authcodee/signfaq.asp#9, pp. 1-13.
Zhang, X. N., "Secure Code Distribution," IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, pp. 76-79, Jun. 1997.
Binstock, Andrew, "Multithreading, Hyper-Threading, Multiprocessing: Now, What's the Difference?," http://www.intel.com/cd/ids/developer/asmo-na/eng/20456.htm, Pacific Data Works, LLC, downloaded Jul. 7, 2008, 7 pp.
Art of Computer Virus Research and Defense by Peter Szor (Feb. 2005).
"Process Execution Controls as a Mechanism to Ensure Consistency," by Eugene Bacic (1990).
"A Flexible Access Control Service for Java Mobile Code," by Corradi, et al., IEEE, pp. 356-365 (2000).
"Microsoft Authenticode Analyzed," by Rohit Khare (Jul. 22, 1996).
"Java Security: Whose Business Is It?," by Mark LaDue (1996).
"Mobile Code Security," by Rubin, et al., IEEE Internet, pp. 30-34, Dec. 1998.
"Protecting Data From Malicious Software," by Schmid, et al., Proceeding of the 18th Annual Computer Security Applications Conference, pp. 1-10, 2002.
"Security in the Large: Is Java's Sandbox Scalable?," by Zhong, et al. (Apr. 1998).
"A Domain and Type Enforcement UNIX Prototype," by Badger, et al., Proceedings of the Fifth USENIX UNIX Security Symposium, Salt Lake City, Utah (Jun. 1995), 15 pp.
"Heuristic Anti-Virus Technology," by Frans Veldman, Aug. 20, 2010.
"Standards for Security in Open Systems," by Warwick Ford (1989).
"Secure File Transfer Over TCP/IP," by Brown, et al., IEEE Region 10 Conference, (Nov. 11-13, 1992).
"Using CASE Tools to Improve the Security of Applications Systems," by Hosmer, et al. (1988).
"Miro: Visual Specification of Security," by Heydon, et al., IEEE Transactions on Software Engineering, vol. 16. No. 10, 13 pp. (Oct. 1990).
"An Evaluation of Object-Based Programming with Visual Basic," by Dukovic, et al. (1995).
"Visual Basic 5.0 Significantly Improved," by W. Dennis Swift, IEEE Antennas and Propagation Magazine, vol. 39, No. 3, 2 pp. (Jun. 1997).
"Development of an Object Oriented Framework for Design and Implementation of Database Powered Distributed Web Applications With the DEMETER Project as a Real-Life Example," by Goschka, et al. (1997).
"Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)," by Anderson, et al. (May 1995).
"A Generic Virus Scanner in C++," by Kumar, et al., Proceedings of the 8th Computer Security Applications Conference, IEEE Press, (Sep. 17, 1992).
"A Model for Detecting the Existence of Software Corruption in Real Time," by Voas, et al., Computers & Security, 12, pp. 275-283 (1993).
"Protection Against Trojan Horses by Source Code Analysis," by Saito, et al., Electronics and Communications in Japan, Part 3, vol. 77, No. 1, 1994, Translated from Denshi Tsushin Gakkai Ronbunshi, vol. 76-A, No. 3, (Mar. 1993), pp. 466-472.
"Information Agents for Automated Browsing," by Dharap, et al. (1996).
"Static Analysis Virus Detection Tools for Unix Systems," by Kerchen, et al., 13th National Computer Security Conference, vol. 1, Washington, DC, Oct. 1-4, 1990, 17 pp.
"Managing Trust in an Information-Labeling System," by Blaze, et al. (Nov. 4, 1996).
"A Guide to the Selection of Anti-Virus Tools and Techniques," by Polk, et al. (Dec. 2, 1992).
"An Integrated Toolkit for Operating System Security," by Rabin, et al., Harvard University Aiken Computation Laboratory Technical Report, 55 pp. (Aug. 1988).
"Intel Launches Virus Counterattack," by Charles Bruno (Aug. 1992).
Intel LANProtect Software User's Guide, Intel Corporation (1992).
"Parents Can Get PC Cruise Control," by George Mannes (Jul. 1996).
"A New Technique for Detecting Polymorphic Computer Viruses," by Carey Nachenberg, Thesis, University of California, Los Angeles (1995).
"Heuristic Scanners: Artificial Intelligence," by Righard Zwienenberg, Virus Bulletin International Conference, 9 pp. (Sep. 20-22, 1995).
Intel LANProtect, 30-Day Test Drive Version User's Manual, 35 pp., 1992.
Earl Boebert's post to the greatcircle firewalls mailing list. Taken from http://www.greatcircle.com/lists/firewalls/archive/firewalls.199410 (Oct. 16, 1994).
CSL Bulletin: "Connecting to the Internet: Security Considerations." Taken from http://csrc.nist.gov/publications/nistbul/cs193-07.txt (Jul. 1993).
FAQ: Interscan Viruswall. Taken from http://web.archive.org/web/19970605050331/www.antivirus.com/faq/finterscanfaq.html (last updated Aug. 8, 1996).
"Network Security and SunScreen SPF-100: Technical White Paper," Sun Microsystems, 1995.
"Why Do We Need Heuristics?," by Frans Veldman, Virus Bulletin Conference, 6 pp. (Sep. 1995).
"Leading Content Security Vendors Announce Support for Check Point Firewall1.3.0; New Partners for Anti-Virus Protection, URL Screening and Java Security," Business Wire, Oct. 7, 1996, available at http://www.allbusiness.com/technology/computernetworks-computer-networksecurity 17274315-1.html#ixzz1gkbKf4g1.
"McAfee Introduces Web Shield; Industry's First Secure Anti-Virus Solution for Network Firewalls; Border Network Technologies and Secure Computing to Enter into Web Shield OEM Agreements," Business Wire, May 14, 1996, available at http://findarticles.comip/articles/mi_mOEINlis_1996_May_14/ai_18283456/.
"Trend Micro Announces Virus and Security Protection for Microsoft Proxy Server; Also Blocks Java Applets, ActiveX," Business Wire, Oct. 29, 1996, available at http://www.thefreelibrary.com/Trend+Micro+announces+virus+and+security+protection+for+Microsoft-aO18810512.
Finjan's Opposition to Websense's Renewed Motion for Judgment as a Matter of Law, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
Declaration of Paul Batcher Re Websense, Inc.'s Proffer of Evidence Re Laches, dated Dec. 19, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
Opposition to Symantec's Motion for JMOL, dated Dec. 17, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
"WWW Proxy to Cut Off Java," by Carl Claunch (Apr. 12, 1996), available at http://groups.google.com/group/muc.lists.firewalls/msg/38bb246b7eb3bec3.
"Combating Viruses Heuristically," by Franz Veldman, Virus Bulletin Conference 67-76 (Sep. 1993).
"Anti-Virus Tools and Techniques for Computer Systems," by Polk, et al., Noyes Data Corporation, Park Ridge, New Jersey, pp. 12-27, Copyright 1995 (with Cover and Copyright Pages—18 pp.).
"Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns," by LeCharlier, et al. (Jul. 2, 1995) (22 pp.).

(56) **References Cited**

OTHER PUBLICATIONS

"Blocking Java Applets at the Firewall," by Martin, et al., 11 pp., Available at http://www.cs.bu.edu/techreports/pdf/1996-026-java-firewalls.pdf (1997).
"Virus: Detection and Elimination," by Rune Skardhamar, Academic Press, Inc., pp. 33-49, Copyright 1996 (with cover and copyright page—18 pp.).
Hruska, Jan, "Computer Viruses and Anti-Virus Warfare (Second Revised Edition)," Ellis Horwood, pp. 87-95, Copyright 1992 (with cover and copyright page—11 p.).
"Active Content Security," by Brady, et al. (Dec. 13, 1999).
"Low Level Security in Java," by Frank Yellin, 13 pp., [retrieved on Dec. 15, 2011], Retrieved from the Internet: mhtml:file//S:\DSFR_DATA/Data03/030052/01_Finjan_v._Symantec_(DDEL-00593)Prio . . .
"Email With a Mind of its Own: The Safe-Tcl Language for Enabled Mail," by Nathaniel Borenstein.
"Mobile Agents: Are They a Good Idea?," by Chess, et al. (Dec. 21, 1994).
"Remote Evaluation," by Stamos, et al., ACM Transaction on Programming Languages and Systems, vol. 12, No. 4 (Oct. 1990).
"Active Message Processing: Messages as Messengers," by John Vittal (1981).
"Programming Languages for Distributed Computing Systems," by Bal, et al., ACM Computing Surveys, vol. 21, No. 3 (Sep. 1989).
"Scripts and Agents: The New Software High Ground," by John Ousterhout (Oct. 20, 1995).
The HotJava Browser: A White Paper, dated Dec. 15, 2011, 5 pp.
The Java Virtual Machine Specification, Sun Microsystems (Aug. 21, 1995).
"Security of Web Browser Scripting Languages: Vulnerabilities, Attacks and Remedies," by Anupam, et al. (Jan. 1998).
"ActiveX and Java: The Next Virus Carriers?," [retrieved on Jan. 2, 2011], 5 pp., Retrieved from the Internet: http://vx.netlux.org/lib/static/vdat/cpactive.htm.
"Gateway Level Corporate Security for the New World of Java and Downloadables" (1996).
"Practical Domain and Type Enforcement for UNIX," by Badger, et al. (1995), 12 pp.
"A Sense of Self for Unix Processes," by Forrest, et al. (1996), 9 pp.
"Antivirus Scanner Analysis 1995," by Marko Helenius, 1995 [retrieved on Dec. 15, 2011], 7 pp., Retrieved from the Internet: http://www.cs.uta.fi/~cschcma/public/vru/documents/test 1995.txt.
"State Transition Analysis: A Rule-Based Intrusion Detection Approach," by Ilgun, et al., IEEE Transactions on Software Engineering, vol. 21, No. 3 (Mar. 1995).
"Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring," by Ko, et al. (1994).
"Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach," by Ko, et al. (1997).
"Classification and Detection of Computer Intrusions," by Sandeep Kumar, Thesis, Purdue University (Aug. 1995).
ThunderByte Anti-Virus Utilities User Manual (1995), 199pp.
Doyle, et al., "Microsoft Press Computer Dictionary," Microsoft Press, 2nd Edition, pp. 137-138, 1993.
"Process Execution Controls: Revisited," by Eugene Bacic (1990).
Schmitt, D.A., ".EXE Files, OS-2 Style," PC Tech Journal, vol. 6, No. 11, p. 76(13), Nov. 1988.
International Search Report for Application No. PCT/IB97/01626, dated May 14, 1998, 3 pp.
Supplementary European Search Report for Application No. EP 97 950351, dated Nov. 17, 2004, 2 pp.
File History for Canadian Application No. 2,275,771, 84 pp.
File History for European Application No. 97950351.3, 58 pp.
File History for Japanese Application No. 10-522345, 48 pp.
Order Construing the Terms of U.S. Pat. Nos. 6,092,194; 6,804,780; 7,058,822; 6,357,010; and 7,185,361, 4 pp., Dec. 11, 2007.
Plaintiff Finjan Software Ltd.'s Opening Claim Construction Brief, 38 pp., Sep. 7, 2007.
Defendant Secure Computing Corporation's Opening Claim Construction Brief, 46 pp., Sep. 7, 2007.
Plaintiff Finjan Software Ltd.'s Answering Claim Construction Brief (Public Version), 45 pp., Sep. 28, 2007.
Defendant Secure Computing Corporation's Responsive Claim Construction Brief (Public Version), 37 pp., Sep. 28, 2007.
Secure Computing Corporation's Disclosure of Prior Art Pursuant to 35 U.S.C. 282, 6 pp., Feb. 1, 2008.
Stang, David J., "Computer Viruses and Artificial Intelligence," Virus Bulletin Conference, pp. 235-257, Sep. 1995.
Johannsen, Magnus, "Java Security and a Firewall Extension for Authenticity Control of Java Applets," Thesis Proposal, Computer Science Department, University of Colorado at Colorado Springs, 5 pp., Jan. 29, 1997.
Joint Appendix of Intrinsic and Extrinsic Evidence Regarding Claim Construction Briefing, vol. 1, Oct. 4, 2007.
Joint Appendix of Intrinsic and Extrinsic Evidence Regarding Claim Construction Briefing, vol. 2, Oct. 4, 2007.
Final Joint Claim Construction Chart, Aug. 24, 2007.
Joint Post-Hearing Claim Construction Chart, Oct. 30, 2007.
Plaintiffs Trial Brief (Public Version), Jan. 14, 2008.
Marcionek, David, "A Complete ActiveX Web Control Tutorial," Available at http://www.codeproject.com/KB/COM/CompleteActiveX.aspx (2006).
Docket for *Finjan Software Ltd.* v. *Secure Computing Corporation, et al.*, U.S. District Court, District of Delaware (Wilmington), Civil Docket No. 1:06-cv-00369-GMS, 52 pp., retrieved on Oct. 24, 2008 from https://ecf.ded.uscourts.gov/cgibin/DktRpt.pl?6532182820_13655-L_567_0-1.
Docket for *Finjan Software Ltd.* v. *Aladdin Knowledge Systems, Inc., et al.*, U.S. District Court, District of Delaware (Wilmington), Civil Docket No. 1:08-cv-00300-GMS, 5 pp., retrieved on Oct. 24, 2008 from https://ecfded.uscourts.gov/cgi-bin-DktRpt.pl?994267838982431-L_567_0-1.
Firewall Toolkit (FWTK) 2.0 Beta Release, 1996.
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., Retrieved from the Internet: http://www.codeguru.com/Cpp/Cpp.cpp_mfc/parsing/article.php/c4093/.
Power, James, "Lexical Analysis," 4 pp., May 14, 2006, Retrieved from the Internet: http://www.cs.may.ie/~jpower/Courses/compilers/notes/lexical.pdf.
Sitaker, Kragen, "Rapid Genetic Evolution of Regular Expressions" [online], The Mail Archive, Apr. 24, 2004 (retrieved on Dec. 7, 2004), 5 pp., Retrieved on the Internet: http://www.mail-archive.com/kragen-tol@canonical.org/msg00097.html.
"Lexical Analysis: DFA Minimization & Wrap Up" [online], Fall, 2004 [retrieved on Mar. 2, 2005], 8 pp., Retrieved from the Internet: http://www.owlnet.rice.edu/~comp412/Lectures/L06LexWrapup4.pdf.
"Minimization of DFA" [online], [retrieved on Dec. 7, 2004], 7 pp., Retrieved from the Internet: http://www.cs.odu.edu/~toida/nerzic/390teched/regular/fa/min-fa.html.
"Algorithm: NFS -> DFA" [online], copyright 1999-2001 [retrieved on Dec. 7, 2004], 4 pp., Retrieved from the Internet: http://rw4.cs.uni-sb.de/ganimal/GANIFA/page16_e.htm.
"CS 3813: Introduction to Formal Languages and Automata—State Minimization and Other Algorithms for Finite Automata," 3 pp., May 11, 2003, Retrieved from the Internet: http://www.cs.msstate.edu/~hansen/classes/3813fall01/slides/06Minimize.pdf.
Watson, Bruce W., "Constructing Minimal Acyclic Deterministic Finite Automata," [retrieved on Mar. 20, 2005], 38 pp., Retrieved from the Internet: http://www.win.tue.nl/~watson/2R870/downloads/madfa_algs.pdf.

(56) **References Cited**

OTHER PUBLICATIONS

Khare, R., "Microsoft Authenticode Analyzed" [online], Jul. 22, 1996 [retrieved on Jun. 25, 2003], 2 pp., Retrieved from the Internet: http://www.xent.com/FoRKarchive/smmer96/0338.html.
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, Retrieved from the Internet: activex.adsp.or.jp/inetsdk/help/overview.htm.
D. Grune, et al., "Parsing Techniques: A Practical Guide," John Wiley & Sons, Inc., New York, New York, USA, pp. 1-326, 2000.
Power, James, "Notes on Formal Language Theory and Parsing," National University of Ireland, pp. 1-40, 1999.
Scott, et al., "Abstracting Application-Level Web Security," ACM, pp. 396-407, 2002.
Non-Final Office Action, dated Jul. 23, 2012, for U.S. Appl. No. 13/290,708, 10 pp.
Amendment and Response to Office Action (dated Jul. 23, 2012) Under 37 C.F.R. 1.111, filed Oct. 23, 2012, for U.S. Appl. No. 13/290,708, 9 pp.
Final Office Action, dated Jan. 7, 2013, for U.S. Appl. No. 13/290,708, 7 pp.
Judgment, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (GMS).
Finjan's Opposition to Sophos' Renewed Motion for Judgment as a Matter of Law, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (GMS).
Finjan's Opposition to Symantec's Motion for Judgment as a Matter of Law at the Close of Evidence, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (OMS).
Chang, Chia-Hsiang, "From Regular Expressions to DFA's Using Compressed NFA's," Oct. 1992, 243 pp., http://www.cs.nyu.edu/web/Research/Theses/chang_chi-hsiang.pdf.
Order Granting/Denying Request for Ex Parte Reexamination for U.S. Appl. No. 90/008,684, dated Aug. 28, 2007, 11 pp.
Request for Ex Parte Reexamination Under 35 U.S.C. 302-307 and 37 C.F.R. 1.510 for U.S. Appl. No. 90/008,684, dated Jun. 6, 2007, 49 pp.
Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,015, filed Oct. 7, 2013,38 pp.
Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,016, filed Oct. 7, 2013,48 pp.
Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,017, filed Oct. 7, 2013, 69 pp.
Notice of Appeal for Civil Action No. 10-593-GMS, filed Sep. 27, 2013, 54 pp.
Order Denying Request for Ex Parte Reexamination, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,015, 16 pp.
Order Granting Request for Ex Parte Reexamination, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,016, 17pp.
"Release Notes for the Microsoft ActiveX Development Kit," Nov. 1996 (10 pp.).
"Part 2: Java Vs. VBS—Breadth of Runtime Environments and Security Make Java a Good Choice for the Internet," Nikkei Electronics, pp. 212-221, No. 658, Mar. 25, 1996.
Lo, Raymond W., et al., "Towards a Testbed for Malicious Code Detection," from the Lawrence Livermore National Laboratory, 9 pp., Feb. 25-Mar. 1, 1991.
"Java Blocking," by C. Claunch, dated Sep. 25, 1996, available at http://groups.google.com/goup/muc.lists.firewalls/msg/2a5ec02c00a37071.
F. M. Avolio and M. J. Ranum, "A Toolkit and Methods for Internet Firewalls," in Technical Summer Conference, pp. 37-44, Boston, Massachusetts, Jun. 1994, USENIX, available at http://citeseer.ist.psu.edu/ranum94toolkit.html.
Petition for Writ of Certiorari to the Supreme Court of the United States in *Finjan, Inc.* v. *United States Patent and Trademark Office*, Apr. 2013.
Brief for the Respondent in Opposition to Petition for Writ of Cert in *Finjan, Inc.* v. *United States Patent and Trademark Office*, Jul. 2013.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Feb. 27, 2009, for U.S. Appl. No. 90/008,678, filed Jun. 6, 2007, 11 pp.
Amendment in Response to Non-Final Office Action dated May 8, 2008, filed Jul. 7, 2008, for U.S. Appl. No. 90/008,678, 27 pp.
Office Action in Ex Parte Reexamination, dated May 8, 2008, for U.S. Appl. No. 90/008,678, 18 pp.
Order Granting/Denying Request for Ex Parte Reexamination for U.S. Appl. No. 90/008,678, dated Aug. 28, 2007, 13 pp.
Request for Ex Parte Reexamination Under 35 U.S.C. 302-307 and 37 C.F.R. 1.510, for U.S. Appl. No. 90/008,678, dated Jun. 6, 2007, 67 pp.
Request for Ex Parte Reexamination Pursuant to 37 C.F.R. 1.501 for U.S. Appl. No. 90/009,175, dated Jun. 6, 2008, 75 pp.
Request for Inter Partes Reexamination Under 35 U.S.C. 311-318 for U.S. Appl. No. 95/001,836, 71 pp., dated Nov. 29, 2011.
Order Construing the Terms of U.S. Pat. Nos. 6,092,194 & 6,480,962, dated Feb. 29, 2012, 4 pp.
File History for U.S. Appl. No. 60/030,639, 33 pp., dated Nov. 8, 1996.
Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, dated Nov. 21, 2011, 246 pp.
Tanenbaum, Andrew S., "Computer Networks," Third Edition, Prentice Hall, Inc., ISBN 0-13-349945-6, Copyright 1996, 9 pp.
Flanagan, David, "Java in a Nutshell, a Desktop Quick Reference," 2nd Edition, O'Reilly & Associates, Inc., ISBN 1-56592-262-X, Copyright 1997, 4 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Proofpoint, Inc. and Armorize Technologies, Inc.*, Case No. 4:13-cv-05808, dated Dec. 16, 2013, 40 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Blue Coat Systems, Inc.*, Case No. 3:13-cv-03999, dated Aug. 28, 2013, 36 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Websense, Inc.*, Case No. 3:13-cv04398, dated Sep. 23, 2013, 27 pp.
Docket for Case No. 3:13-cv-03999, *Finjan, Inc.* v. *Blue Coat Systems, Inc.*, Updated Dec. 5, 2013, 9 pp.
Docket for Case No. 3:13-cv-04398, *Finjan, Inc.* v. *Websense, Inc.*, Updated Nov. 25, 2013, 7 pp.
Zhong, et al., "Security in the Large: Is Java's Sandbox Scalable?," Seventh IEEE Symposium on Reliable Distributed Systems, pp. 1-6, Oct. 1998.
Hopwood, David, "A Comparison Between Java and ActiveX Security," Network Security, Elsevier Science Ltd., pp. 15-20, Dec. 1997.
Ungureanu, Victoria, et al., "A Policy-Based Access Control Mechanism for the Corporate Web," 16th Annual Computer Security Applications Conference (ACSAC '00), IEEE, Dec. 11, 2000.
Ham, Lein, et al., "A Software Authentication System for Information Integrity," Computers & Security, 1992.
Pei, Cao, Jin, et al., "Active Cache: Caching Dynamic Contents on the Web," Middleware '98, Springer London, Mar. 1998.
Bharadvaj, Harini, et al., "An Active Transcoding Proxy to Support Mobile Web Access," Dept. of Computer Engineering & Computer Science, University of Missouri-Columbia, Nov. 1998.
Bharadvaj, Harini, et al., "An Active Transcoding Proxy to Support Mobile Web Access," Proceedings, Seventeenth IEEE Symposium, IEEE, 1998.
Chi, Chi-Hung and Wu, Yin, "An XML-Based Data Integrity Service Model for Web Intermediaries," Proceedings, 7th IWCW, Aug. 14, 2002.
Article, "Java: The Inside Story, We Interview Java's Creators to Find What They Had in Mind," Sun World Online, Jul. 1995.
Giles, James, et al., "Authentication for Distributed Web Caches," Computer Security—ESORICS 2002, Springer Berlin, Heidelberg, Oct. 16, 2002.
Mandar, Raje, "Behavior-Based Confinement of Untrusted Applications," Dept. of Computer Science, UC Santa Barbara, TRCS 99-12, pp. 1-96, Jan. 1999.
Check Point Software Technologies Ltd., Archived webpages regarding Content Security and Partners, Check Point Software, 1997.

(56) **References Cited**

OTHER PUBLICATIONS

Hassler, Vesna and Then, Oliver, "Controlling Applets' Behavior in a Browser," Computer Security Applications Conference, Proceedings of the 14th Annual IEEE.

Seltzer, Margo I., et al., "Dealing With Disaster: Surviving Misbehaved Kernel Extensions," Proceedings of the USENIX 2nd Symposium on Operating Systems Design and Implementation, Seattle Washington, Oct. 1996.

Golvers, Luc, "Delivery and Installation of Software: Disputes and the Burden of Proof, Information Systems Security," Chapman & Hall, May 31, 1996.

Emin Gün Sirer, et al., "Design and Implementation of a Distributed Virtual Machine for Networked Computers," Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles, pp. 202-216, Dec. 5, 1999.

Dr. Solomon's Anti-Virus Toolkit (prerelease), PC Magazine, Nov. 21, 1995.

"Draft, Internet Component Download," Microsoft Corporation, Jan. 1996.

Evans, David and Twyman, Andrew, "Flexible Policy-Directed Code Safety," Proceedings of the 1999 IEEE Symposium on Security and Privacy (IEEE), May 12, 1999.

Jackson, Keith, "FluShot+," Virus Bulletin, Nov. 1989.

"HotJava Browser Version 1.0 Frequently Asked Questions," Sun Microsystems, Inc., Nov. 5, 1997.

"HotJava Browser Version 1.0 Website Product Information," Sun Microsystems, Inc., Jan. 12, 1998.

"HotJava Browser: A White Paper," Sun Microsystems, Inc., 1994.

"IBM WebSphere Edge Server for Multiplatforms Administration Guide, Version 2.0," IBM, Dec. 2001.

Braswell, Byron, et al., "IBM WebSphere Edge Server: New Features and Functions in Version 2, IBM Redbooks," IBM, Apr. 2002.

Chockler, Gregory V., et al., "Implementing a Caching Service for Distributed CORBA Objects," Middleware 2000, Springer Berlin, Heidelberg, Apr. 4, 2000.

Claunch, Carl V., "Improved JavaScript and Java Screening Function," May 4, 1996.

"Integralis Announces MIMEsweeper for NT Version 2.1," Business Wire, Feb. 20, 1996.

"Integralis Launches the Internet's First Anti-Virus Protection System," M2 Press WIRE, Jun. 30, 1995.

"Integralis Mimesweeper Scans Internet Mail for Viri Nasties," Computer Business Review, Jul. 18, 1995.

Ethington, Brent, "Introducing Microsoft Windows 95," Microsoft Press, 1995.

Shin, Insik and Mitchell, John C., "Java Bytecode Modification and Applet Security," Stanford CS Tech Report, 1998.

McGraw, Gary and Felten, Edward W., "Java Security: Hostile Applets, Holes, and Antidotes," John Wiley & Sons, Inc., 1997.

"JavaSoft JAR File Format Specification Version 1.0," Sun Microsystems, Inc., Aug. 8, 1996.

"Jingle Bells, Jingle Bells," Virus Bulletin, Jan. 1996.

Greenberg, Ross M., "Know Thy Viral Enemy," BYTE, Jun. 1989.

Vigil, Henry, "Making the Internet Safe for E-Commerce," Datamation, Oct. 1996.

Sibert, W. Olin, "Malicious Data and Computer Security," National Information Systems Security Conference, Oct. 22, 1996.

"MIMEsweeper Administrator Guide," Integralis, 1995.

"MIMEsweeper Review, PC User," PC User, Oct. 1996.

"MIMESweeper Review, Secure Computing," Secure Computing, Sep. 1996.

Small, Christopher, "MiSFIT: A Freely Available Tool for Building Safe Extensible Systems," Harvard Computer Science Technical Reports, Jul. 1996.

Tardo, Joseph and Valente, Luis, "Mobile Agent Security and Telescript," Compcon '96, 'Technologies for the Information Superhighway' Digest of Papers (IEEE), Feb. 25, 1996.

Chander, Ajay, et al., "Mobile Code Security by Java Bytecode Instrumentation," Computer Science Dept., Stanford University, and Dept. of Computer and Information Science, University of Pennsylvania.

Greenberg, Ross M., "MS-DOS Anti-Virus Tools and Techniques or How I Stopped Worrying and Learned to Trust in Debuggers," Virus Bulletin Conference, Sep. 1991.

Cheng, Kai and Kambayashi, Yahiko, "Multicache-Based Content Management for Web Caching," Proceedings of the First International Conference on Web Information Systems Engineering, Nov. 2000.

"Netscape Proxy Server, Chapter 9: How Caching Works," Netscape Communications Corporation, 2000.

Stallings, William, "Network and Internetwork Security Principles and Practice," Prentice Hall, 1995.

Weaver, Alfred C., et al., "Network Edge Services," 28th Annual Conference of the Industrial Electronics Society, IEEE.

Norton AntiVirus for Windows 95, PC Magazine, Nov. 21, 1995.

Anupan, Joshi, "On Proxy Agents, Mobility, and Web Access," Mobile Networks and Applications 5, pp. 233-241, 2000.

Riggs, Roger, et al., "Pickling State in the Java System," Proceedings of the USENIX 1996 Conference on Object-Oriented Technologies, Toronto, Ontario, Canada, Jun. 1996.

Jaeger, Trent and Rubin, Aviel D., "Preserving Integrity in Remote File Location and Retrieval," Proceedings of the Symposium on Network and Distributed System Security (IEEE), Feb. 22, 1996.

Press Release: "Trend Micro Ships InterScan AppletTrap—Enterprise Java and ActiveX Security Software," Business Wire, Mar. 23, 1999.

Thorn, Tommy, "Programming Languages for Mobile Code," ACM Computing Surveys, vol. 29, No. 3, pp. 213-239, 1997.

Necula, George C., "Proof-Carrying Code, Design and Implementation," School of Computer Science, Carnegie Mellon University, Sep. 1996.

Pandey, Raju and Hashii, Brant, "Providing Fine-Grained Access Control for Mobile Programs Through Binary Editing," TR-98-08, California University, Davis Dept. of Computer Science, 1997.

Iglio, Pietro, et al., "Rule-Based Filtering for Java Applets," Proceedings of the 14th Annual Security Applications Conference (IEEE), Dec. 1998.

Iglio, Pietro, et al., "Rule-Based Filtering for Java Applets," Computer Security Applications Conference, Proceedings of the 14th Annual IEEE, Dec. 1998.

Kirtland, Mary, "Safe Web Surfing With the Internet Component Download Service," Microsoft Systems Journal, Jul. 1996.

Erlingsson, Ulfar and Scheider, Fred B., "SASI Enforcement of Security Policies: A Retrospective," Proceedings of the 1999 Workshop on New Security Paradigms, ACM, 1999, Sep. 23, 1999.

Rubin, Aviel D., "Secure Distribution of Electronic Documents in a Hostile Environment," Computer Communications, Jun. 6, 1995.

Hsiao, Hui-I. and Brannon, Karen W., "Secure Information Caching on the Web," Advanced Web Technologies and Applications, Springer Berlin Heidelberg, 2004, Apr. 14, 2004.

Herbert, Andrew, "Secure Mobile Code Management Enabling Java for the Enterprise," Digitivity, Inc., May 1997.

Rosenblit, Moshe, "Secure Software Distribution," Network Operations and Management Symposium, 1994 (IEEE), Feb. 14, 1994.

Hashi, Brant, et al., "Securing Systems Against External Programs," Internet Computing IEEE, Nov. 1998.

Farmer, William M., et al., "Security for Mobile Agents: Authentication and State Appraisal," Computer Security—ESORICS 96, Springer Berlin Heidelberg, 1996, Sep. 25, 1996.

Wakeman, Ian, et al., "Signaling in a Component Based World," 1998 IEEE, 1998.

"Sun Microsystem's JavaSoft Java Beans 1.0 API Specification," Sun Microsystems, Inc., Oct. 15, 1996.

"Sweep Website Product Overview—Client/Server Virus Detection: A Corporate Solution," Sophos, Dec. 11, 1997.

"Sweep Website Product Overview—SWEEP Virus Detection for Windows NT," Sophos, Feb. 10, 1998.

"Symantec Announces First Native Java Virus Scanner and Leading Edge Virus Analysis Technology," Symantec Corp., Mar. 27, 1996.

(56) **References Cited**

OTHER PUBLICATIONS

Malkhi, Dahlia, et al., "The Design and Implementation of a Java Playground," AT&T Labs—Research, Oct. 10, 1997.
Sirer, Emin Gün, "The Kimerawork on Bytecode Verification," Department of Computer Science and Engineering, University of Washington, Internet: http://kimera.cs.washington.edu/overview.html, Oct. 30, 1997.
Evans, David E., "Thesis by David E. Evans, Policy-Directed Code Safety," Massachusetts Institute of Technology, Feb. 2000.
"ThunderBYTE Anti-Virus Utilities for Windows 95 (prerelease)," PC Magazine, Nov. 21, 1995.
Iliadis, J., et al., "Towards Secure Downloadable Executable Content: The Java Paradigm," Berlin Heidelberg, 1998.
Maheshwari, Anuj, et al., ""TranSquid: Transcoding and Caching Proxy for Heterogenous Ecommerce Environments," Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems," 2000.
"Trend Micro Product Paper, InterScan AppletTrap Eliminates Malicious Mobile Code," Trend Micro, Inc., Sep. 1999.
Swamp, Vipin, "Trust Appraisal and Secure Routing of Mobile Agents," DARPA Workshop on Foundations for Secure Mobile Code, Monterey, CA, USA, Feb. 17, 1997.
Feigenbaum, Joan and Lee, Peter, "Trust Management and Proof-Carrying Code in Secure Mobile-Code Applications," DARPA Workshop on Foundations for Secure Mobile Code, vol. 26, Mar. 27, 1997.
Chu, Yang-Hua, "Trust Management for the World Wide Web," Diss. Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Jun. 13, 1997.
Rabinovich, Michael and Spatscheck, Oliver, "WEB Caching and Replication," Addison Wesley, 2002.
Krishnamurthy, Balachander and Rexford, Jennifer, "Webware: Nightmare or Dream Come True?", National Information Systems Security Conference, Oct. 22, 1996.
"White Paper: HotJava Views 1.1 Software System Overview and Applet Integration," Sun Microsystems, Inc., Jun. 16, 2000.
Greer, Earl, "Wise Dr. Solomon's Fast Accurate in Catching Viruses," InfoWorld, Nov. 6, 1995.
Ghosh, Anup K., et al., "Wrapping Windows NT Software for Robustness," Digest of Papers, Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (IEEE), Jun. 18, 1999.
Docket for *Finjan, Inc.* v. *Fireeye, Inc.*, Case No. 4:13-cv-03133-SBA, dated Jul. 8, 2013.
Docket for *Finjan, Inc.* v. *Websense, Inc.*, Case No. 3:13-cv-04398-JSW, dated Sep. 23, 2013.
Twyman, Andrew R., "Flexible Code Safety for Win32," May 21, 1999.
Wagner, David A., "Janus: An Approach for Confinement of Untrusted Applications," 1999.
Symantec, "The Digital Immune System: Enterprise-Grade Anti-Virus Automation in the 21st Century," 2001.
Aladdin Knowledge Systems White Paper, "Complete Content Security for Internet Gateways and Mail Servers," eAladdin.com, 2002.
Ositis, Inc., "Ositis Announces eShield Platform," Business Wire Press Release, May 6, 2003.
Chen, "Poison Java," Aug. 1999.
Aladdin Knowledge Systems, Inc., "Privilege Secure Software Distribution: Comprehensive Electronic Software Licensing and Distribution Solution," Sep. 1999.
"EliaShim Announces Release of Unique Anti Virus Plug-In for Checkpoint's Firewall-1," PRNewswire Press Release, Feb. 10, 1997.
Wagner, David, et al., "A First Step Toward Automated Detection of Buffer Overrun Vulnerabilities," Feb. 3, 2000.
Bishop, Matt, "A Taxonomy of UNIX System and Network Vulnerabilities," May 1995.
Aladdin Knowledge System, "Aladdin eSafe Protect Desktop User's Manual," 1999.
Necula, George Ciprian, "Compiling With Proofs," Sep. 18, 1998.
Lambright, H. Dan., "Automated Verification of Mobile Code," Nov. 3, 1997.
Meyers, et al., "Automatic Detection of C++ Programming Errors: Initial Thoughts on a Lint ++," Aug. 1991.
Bishop, Matt and Dilger, Michael, "Checking for Race Conditions in File Access," Sep. 1995.
Aho, Alfred V., et al., "Compilers: Principles, Techniques, and Tools," Mar. 1988.
Kozen, Dexter, "Efficient Code Certification," Jan. 8, 1998.
Varvitsiotis, Angelos, et al., "EuroMed-Java: Trusted Third Party Services for Securing Medical Java Applets," Sep. 16, 1998.
Detlefs, David L., et al., "Extended Static Checking," Dec. 18, 1998.
Data Fellows Corp., "F-Secure Script Viruses Detector and Eliminator, Version 1.6," 1998.
Morrisett, Greg, et al., "From System F to Typed Assembly Language (Extended Version)," Nov. 21, 1997.
Reps, Thomas, et al., "Incremental Context-Dependent Analysis for Language-Based Editors," 1983.
Viega, John, et al., "ITS4: A Static Vulnerability Scanner for C and C++ Code," Dec. 11, 2000.
Symantec, "Norton Antivirus for Firewalls 1.5 Eliminates Internet Viruses on IBM, CVP Firewalls Before They Enter Corporate Network," Oct. 20, 1999.
Johnson, S.C., "Lint, a C Program Checker," Jul. 26, 1978.
Ghosh, Anup K., "On Certifying Mobile Code for Secure Applications," Nov. 4, 1998.
Aladdin Knowledge System, "Safe Internet Connectivity for Home and Small Office," May 1999.
Bergeron, J., et al., "Static Analysis of Binary Code to Isolate Malicious Behaviors," Jun. 16, 1999.
Larochelle, David, et al., "Statically Detecting Likely Buffer Overflow Vulnerabilities," Aug. 13, 2001.
Radatti, Peter V., "The CyberSoft Virus Description Language," Feb. 1996.
Levy, Jacob Y., et al., "The Safe-Tcl Security Model," Jun. 1998.
Kipelainen, Pekka, "Tree Matching Problems With Applications to Structured Text Databases," Nov. 1992.
Trend Micro, "Trend InterScan WebManager 1.1," 1999.
McGraw, Gary, et al., "Understanding the Keys to Java Security—The Sandbox and Authentication," May 1, 1997.
Bishop, Matt, "Vulnerabilities Analysis," Sep. 1999.
Aladdin Knowledge System, "Web Archive eSafe.com (eSafe Protect Gateway)," Oct. 13, 1999.
Aladdin Knowledge Systems, "Web Archive of eSafe.com," Mar. 2000.
Bailey, "C ++ SoftBench CodeAdvisor 5.0," Dec. 2005.
Ribera, et al., "An Evaluation of Tools for Static Checking of C ++ Code," Feb. 2000.
ParaSoft, "Code Wizard," Nov. 1999.
Wilson, "A Look at PREfix by Intrinsa," Software Testing & Quality Engineering, May/Jun. 1999.
Internet Archive for www.prolint.com, "Revision History," Apr. 11, 2003.
Ghosh, et al., "An Automated Approach for Identifying Potential Vulnerabilities in Software," May 1998.
Internet Archive, "Secure Software Solutions: RATS," Jun. 6, 2001.
Christodorescu, Mihai and Jha, Somesh, "Static Analysis of Executables to Detect Malicious Patterns," Proceedings of the 12th USENIX Security Symposium, Aug. 4-8, 2003.
Roesch, Martin and Green, Chris, "Snort User Manual, Snort Release: 1.9.1," Feb. 26, 2003.
Kephart, J., et al., A Biologically Inspired Immune System for Computers, 1994.
Kephart, J., et al., Automatic Extraction of Computer Virus Signatures (1994).
Kephart, J., et al., Biologically Inspired Defenses Against Computer Viruses (1995).
S&S International, Dr. Solomon's Anti-Virus Toolkit for Windows and DOS (1995).
A. Solomon, Dr. Solomon's Virus Encyclopedia (1995).
Dr. Solomon's Anti-Virus Toolkit for Windows and DOS.
Dr. Solomon's Anti-Virus Toolkit for Netware.
Lunt, et al., "A Real-Time Intrusion Detection Expert System (IDES)," 1992.
"Sweep Windows NT User Manual," 1996.

(56) **References Cited**

OTHER PUBLICATIONS

"Sweep User Manual for Windows NT," 1995.
"Sweep User Manual for Windows 95," 1995.
Smaha, S., "Haystack: An Intrusion Detection System" (1988).
Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria" (1985).
"Check Point Software Delivers Breakthrough Security Advancements with Firewall-1 3.0; Industry's First to Offer Content Security Connection Control, Advanced Authentication and Encryption Integrated into an Open Enterprise Security Platform." Business Wire, 1996.
Venema, W., "TCP WRAPPER Network Monitoring, Access Control, and Booby Traps," in Proceedings of the Third Usenix UNIX Security Symposium (1992).
Radatti, "Computer Viruses in Unix Networks" (1995).
"Changing the Rules on Viruses," PC Magazine, Aug. 1994.
Chen, Steven, "PC Rx," PC Magazine, Mar. 16, 1993.
Sophos, "Intercheck," 1995.
Virus Bulletin, 20 pp., Nov. 1989.
Virus Bulletin, 24 pp., Aug. 1996.
Virus Bulletin, 32 pp., Jun. 1996.
Virus Bulletin, 24 pp., May 1995.
Virus Bulletin, 24 pp., Mar. 1995.
Virus Bulletin, 24 pp., Jun. 1995.
Virus Bulletin, 24 pp., Aug. 1995.
Virus Bulletin, 24 pp., Sep. 1995.
Virus Bulletin, 24 pp., May 1994.
Virus Bulletin, 24 pp., Mar. 1994.
Virus Bulletin, 24 pp., Feb. 1994.
Virus Bulletin, 24 pp., Apr. 1994.
Virus Bulletin, 24 pp., Jul. 1995.
Virus Bulletin, 24 pp., Apr. 1995.
Virus Bulletin, 24 pp., Dec. 1996.
Virus Bulletin, 24 pp., Oct. 1995.
Virus Bulletin, 24 pp., Jan. 1994.
Virus Bulletin, 24 pp., Jul. 1994.
Virus Bulletin, 24 pp., Aug. 1994.
Virus Bulletin, 24 pp., Feb. 1996.
Virus Bulletin, 24 pp., Feb. 1995.
Virus Bulletin, 24 pp., Nov. 1994.
Virus Bulletin, 24 pp., Nov. 1995.
Virus Bulletin, 24 pp., Nov. 1996.
Virus Bulletin, 24 pp., Dec. 1995.
Virus Bulletin, 24 pp., Dec. 1994.
Virus Bulletin, 24 pp., Oct. 1994.
Virus Bulletin, 24 pp., Apr. 1996.
Virus Bulletin, 24 pp., Sep. 1994.
Virus Bulletin, 24 pp., Jun. 1994.
Virus Bulletin, 24 pp., Sep. 1996.
Virus Bulletin, 24 pp., Jan. 1996.
Virus Bulletin, 24 pp., Mar. 1996.
Virus Bulletin, 24 pp., Jan. 1995.
Virus Bulletin, 24 pp., May 1996.
Virus Bulletin, 28 pp., Jul. 1996.
Virus Bulletin, 24 pp., Oct. 1996.
U.S. Appl. No. 60/711,012, filed Aug. 2005.
U.S. Appl. No. 60/711,972, filed Aug. 2005.
U.S. Appl. No. 60/735,772, filed Nov. 10, 2005.
U.S. Appl. No. 60/581,857, filed Jun. 21, 2004.
Gene H. Kim and Eugene H. Spafford, "The Design and Implementation of Tripwire: A File System Integrity Checker," 23 pp., 1993.
Mark Kennedy, "Script-Based Mobile Threats," Virus Bulletin Conference, Sep. 2000.
Dr. Vesselin Bontchev and Katrin Tocheva, "Macro and Script Virus Polymorphism," Proceedings of the Twelfth International Virus Bulletin Conference, pp. 406-437, 2002.
Chris Shiflett, "PHP Security," O'Reilly Open Source Convention, Jul. 26, 2004.
"Mod-Security Reference Manual, v. 1.8.5," Oct. 26, 2004.
Belden Menkus, "A Secure Electronic Document Audit Trail Product," Jun. 1995.
David R. Safford, Douglas Lee Schales, and David K. Hess, "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment," Proceedings of the Fourth USENIX UNIX Security Symposium, Santa Clara, California, Oct. 4-6, 1993.
F-Prot, "Our Technology—F-Prot Antivirus" [online], Copyright 2014 [retrieved on Jun. 7, 2015], Retrieved from the Internet: http://www.f-prot.com/technology/.
"Parasoft Introduces CodeWizard" [online], May 5, 1996 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/430/.
"CodeWizard for Windows 95 and Windows NT Available Soon!" [online], Oct. 1, 1996 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/429/.
"Parasoft Announces CodeWizard for Windows" [online], Mar. 3, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/428/.
"CodeWizard Supports Microsoft Visual C++ v. 5.0" [online], Mar. 24, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/426/.
"Parasoft Releases Preliminary Version of CodeWizard for Java" [online], Mar. 24, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/427/.
"Parasoft Releases CodeWizard v. 1.1 Beta" [online], May 23, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/425/.
"Parasoft Announces Pre-Release Version of CodeWizard for Java for Windows NT/95" [online], Jun. 2, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/424/.
"Parasoft Releases CodeWizard v. 1.1" [online], Jul. 3, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/423/.
"Parasoft Announces CodeWizard v. 1.5, Preventative Medicine for Error-Prone Code" [online], Jul. 28, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/422/.
"CodeWizard—Parasoft Forum" [online], Apr. 27, 2006 (downloaded May 8, 2015), Retrieved From: http://forums.parasoft.com/index.php?showtopic=996&mode=threaded&pid=2560.
"Parasoft Introduces CodeWizard Plus the Intelligent Development Solution for C++ and Java Developers" [online], Sep. 2, 1997 (downloaded May 7, 2015), Retrieved From: http://www.parasoft.com/press/421/.
"Parasoft Launches Newest Version of Award-Winning Automatic Error Detection Tool for C/C++, Insure++, v. 4.1" [online], Jun. 8, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/315/.
"Parasoft Awarded Logo Trademark for C/C++ Test Coverage Tool, TCA" [online], Aug. 31, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/313/.
"Parasoft Tools Insure++ and CodeWizard Offer Needed Stability, Reliability for Microsoft Visual C++ 6.0 Developers" [online], Oct. 28, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/311/.
"Parasoft's Insure++ and CodeWizard Now Support 64-Bit Development on HP11 and SGI64" [online], Dec. 7, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/308/.
"Newly Enhanced Versions of CodeWizard for C++ and Java Coming Soon" [online], Nov. 4, 1997 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/420/.
"Parasoft Releases Beta Version of CodeWizard for HTML, a Web Error Prevention Tool for Use With WebKing" [online], Feb. 1, 1999, [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/416/.
"Internet Error Prevention and Detection: How Dysfunctional Are the Fortune 100 Web Sites?" [online], Nov. 8, 1999 [retrieved on May 8, 2015], Retrieved From the Internet: .1,zio. \.\-. . . .s. .:,i.0 il ' „ 73Q.
"Parasoft Director of Technology, Dr. Michael Aivazis, to Speak at LinuxWorld Conference and Expo" [online], Jan. 24, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/655/.

(56) **References Cited**

OTHER PUBLICATIONS

"Parasoft Presents Solutions for Object-Oriented Embedded Development" [online], Feb. 21, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/654/.
"Parasoft to Showcase Solutions for Web Development at Internet World" [online], Mar. 27, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/735/.
"New Versions of CodeWizard for C++ and Java Available Now!" [online], Dec. 4, 1997 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/419.
Adam Kolawa, Ph.D., "Parasoft & Innovation" [online], Apr. 2008 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/company/20-year-retrospective/.
"Parasoft Extends Support to Windows 2000" [online], Apr. 10, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/645/.
"Boeing Chooses Parasoft's CodeWizard to Automate Coding Standard" [online], Apr. 17, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/414/.
"Parasoft and DCS IP, LLC Announce Sales Partnership" [online], May 2, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/572/.
"Parasoft Releases CodeWizard v 3.1 for C and C++" [online], May 3, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/413/.
"Parasoft to Present Newest Upgrade of WebKing at WebLA" [online], May 15, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/733/.
"Parasoft Releases CodeWizard Version 3.2" [online], Sep. 25, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/412/.
"Parasoft Releases New Version of Java Tool CodeWizard for Java v. 1.1" [online], May 26, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/418/.
"IMA Reduces the Time and Cost of Regulatory Compliance Through Static Analysis," Parasoft, 2 pp.
"Parasoft Forum" [online], Apr. 8, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showtopic=206&mode=threaded&pid=415.
"Parasoft C/C++ Test" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/product/cpptest/.
"Parasoft, This Forum" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showforum=16.
"Parasoft Forum, Website" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showforum=17.
"Parasoft Forum, Custom Link Check" [online], Jun. 22, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t275.html.
"Parasoft Forum, How Can I Check My Codes Without Compiler" [online], Jun. 9, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t264.html.
"Parasoft Forum, CodeWizard Installation Problem" [online], Aug. 1, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion.pho/t306.html.
"Parasoft Launches Newest Version of Award-Winning Automatic Software Detection Tool for C/C++, Insure++ v. 4.1 for Windows" [online], Jun. 22, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/314/.
"Parasoft's CodeWizard for Java Avoid "Nasty" Surprises in Their Programs" [online], Sep. 21, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/417/.
"Parasoft Forum, CodeWizard" [online], Apr. 27, 2006 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t996.html.
"Parasoft Forum, WebKing and Usability Test" [online], Feb. 23, 2005 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t500.html.
"Parasoft Forum, Repeated Element Regression Control" [online], Jun. 9, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showtopic=262.
"Parasoft Sitemap" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/sitemap.
"Parasoft to Exhibit Solutions for Object-Oriented Embedded Development at Embedded Systems Conference, Boston" [online], Jun. 26, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/638/.
"Parasoft to Showcase Error Prevention and Error Detection Software at Upcoming Embedded Systems Conference" [online], Sep. 18, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/628/.
"Parasoft Launches New Automated Database Testing Tool to Validate Application-Database Interaction" [online], Nov. 19, 2002 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1010/.
"Parasoft Joins Eclipse" [online], Dec. 10, 2002 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1023/.
"Parasoft WebKing Earns IBM Ready for WebSphere Validation" [online], Jan. 29, 2003 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1033/.
"Top Companies Choose Parasoft to Improve Software Quality and Reliability" [online], Jul. 2, 2003 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1033/.
"Parasoft Extends Support to Software Developers in the Automotive Industry" [online], Jan. 7, 2004 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1336/.
"Parasoft Forum, ANT Task for CodeWizard" [online], Aug. 12, 2004 [retrieved on May 8, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=319.
"Parasoft's CodeWizard v. 4.0 Enhances Software Error Prevention Capabilities" [online], Aug. 27, 2001 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/411/.
Arthur Hicken, "Service Virtualization, Development Resting, API Testing Tips, False Positives in Statuc Code Analysis" [online], Feb. 21, 2013 [retrieved on May 8, 2015], Retrieved from the Internet: http://blog.parasoft.com/bid/113524/False-Positive-in-Static-Code-Analysis.
"Parasoft Forum, CodeWizard Forum, Where to Post CodeWizard Topics?" [online], Apr. 6, 2004 [retrieved on May 7, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=201.
"CodeWizard User Testimonial (Suzette LaGray)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/346/.
"CodeWizard User Testimonial (Mike LeBlanc, SKY Computers, Inc.)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/348/.
"CodeWizard User Testimonial (Javier Borrajo, Telefonica Investigacion & Desarolio)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/351/.
"CodeWizard User Testimonial (Gura Rao, Data Research Associates)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/353/.
"CodeWizard User Testimonial (Martin A. Bailey, Richardson Communications Group)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/355/.
"Parasoft Forum, Download CodeWizard" [online], Jul. 25, 2005 [retrieved on May 8, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=682.
"Parasoft, Awards & Recognitions" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/company/awards/.

(56) **References Cited**

OTHER PUBLICATIONS

Parasoft, "Software Testing Tools—From Static Analysis to Service Virtualization" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/products/.

"Parasoft CodeWizard" [online], Copyright 1996-2001 [retrieved on Jun. 5, 2015], Retrieved from the Internet: http://char.tuiasi.ro/doace/www.parasoft.com/products/wizard.

Ben Crowder, "CodeWizard for Linux" [online], Nov. 1, 1999 [retrieved on Jun. 5, 2015, Linux Journal, Retrieved from the Internet: http://www.linuxjournal.com/article/3615.

Tom Bragg, "Parasoft's CodeWizard for Java" [online], Dec. 18, 1999 [retrieved on Jun. 5, 2015], Retrieved from the Internet: http://adtmag.com/articles/1999/12/18/parasofts-codewizard-for-java.aspx.

U.S. Pat. No. 6,154,844 Prosecution History Excerpts.

Marc Record ("A Flexible Security Model for Using Internet Content"), 1997.

Joint Claim Construction and Pre-Hearing Statement Pursuant to Local Rule 4-3, dated Mar. 16, 2015, submitted by Finjan, Inc. and Symantec Corporation, Case No. 3-14-CV-02998-HSG.

Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, dated Apr. 20, 2015, 30 pp., Case No. 14-CV-02998-HSG.

Wahbe, et al., "Efficient Software-Based Fault Isolation," ACM SIGOPS (Special Interest Group on Operating Systems) Symposium on Operating Systems Principles, Dec. 1993.

Jack W. Davidson, et al., "Safe Virtual Execution Using Software Dynamic Translation," 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, Dec. 2002.

Random House Webster's College Dictionary, Second Edition, Random House, New York, Copyright 1999, Cover, Copyright Page and p. 339 (Database).

Webster's New World Dictionary of Computer Terms, Fourth Edition, Prentice Hall, New York, NY, Copyright 1992, and p. 95 (Database).

U.S. Appl. No. 60/205,591, filed May 17, 2000, 9 pp.

Reexamination Excerpts for U.S. Appl. No. 90/013,017, 9 pp.

File History Excerpts from U.S. Appl. No. 13/290,708 (Now U.S. Pat. No. 8,677,494), 29 pp.

U.S. Appl. No. 12/174,592 (Petition to Accept Unintentionally Delayed Claim of Priority filed on Dec. 20, 2011 (denied) and Jan. 25, 2012 (granted).

U.S. Appl. No. 12/174,592 (applicants' response and amendment filed on Jul. 19, 2011).

MARC Record ("A Sense of Self for Unix Processes"), 1996.

Virus Bulletin Conference, Proceedings of the Fifth International Virus Bulletin Conference, Sep. 1995 (Parts 1-3).

MARC Record ("Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns"), 1995.

Duncan, Ray, "Advanced MS DOS," Microsoft Press, Redmond Washington, Copyright 1986, cover, Copyright Page, and pp. 272-282.

"Digital Signature Initiative Proposal," Aug. 1, 1996.

James W. Moore, "The Use of Encryption to Ensure the Integrity of Reusable Software Components," IEEE, 1994.

Judson D. Weeks, et al., "CCI-Based Web Security: A Design Using PGP," W3C, Dec. 13, 1995.

Burkhard Wiegel, "Secure External References in Multimedia Email Messages," Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, Mar. 14-15, 1996.

Ofer Maor, et al., "SQL Injection Signatures Evasion," Imperva Application Defense Center, Apr. 2004.

Ryoichi Sasaki, et al., "Security and Privacy in the Age of Ubiquitous Computing," IFIP TC11 20th International Information Security Conference, Chiba, Japan, May 30-Jun. 1, 2005.

Francisco Fernandez, "Heuristic Engines," Virus Bulletin Conference, Sep. 2001.

Alex Shipp, "Heuristic Detection of Viruses Within Email," Virus Bulletin Conference, Sep. 2001.

Klaus Julisch, et al., "Detection of Intrusions and Malware, and Vulnerability Assessment," Second International Conference, DIMVA 2005, Vienna, Austria, Jul. 7-8, 2005.

Christopher Kruegel, et al., "On the Detection of Anomalous System Call Arguments," Computer Security—ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjovik, Norway, Oct. 13-15, 2003.

John Graham-Cumming, "The Waxing and Waning of Spammers' Trickery," Proceedings of the 14th Virus Bulletin International Conference, Chicago, Sep. 29-Oct. 1, 2004.

Eric Lippert's Blog, "Eval is Evil, Part Two," Nov. 4, 2003.

* cited by examiner

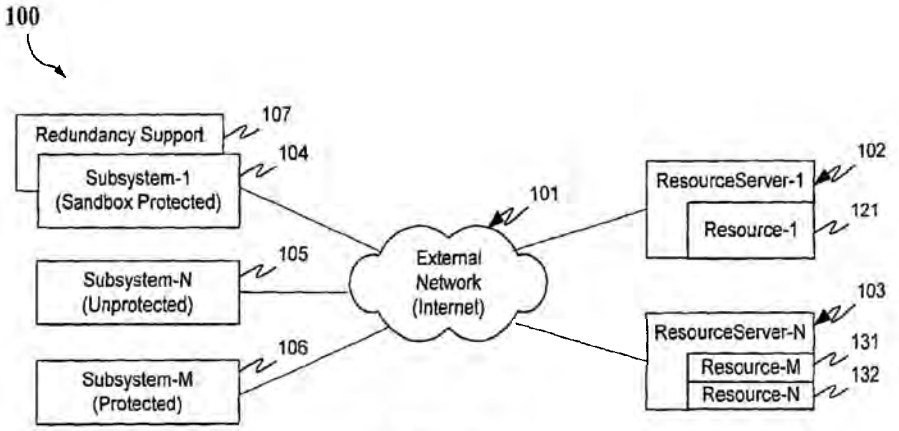


FIG. 1a

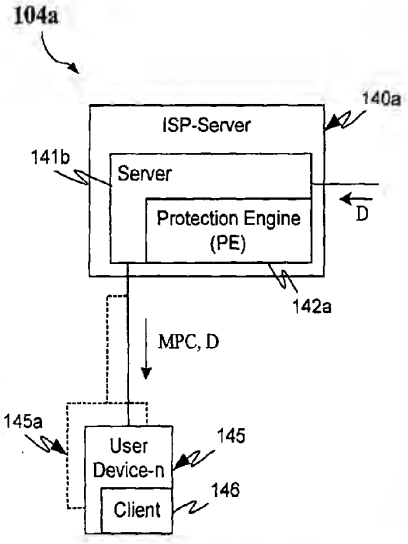


FIG. 1b

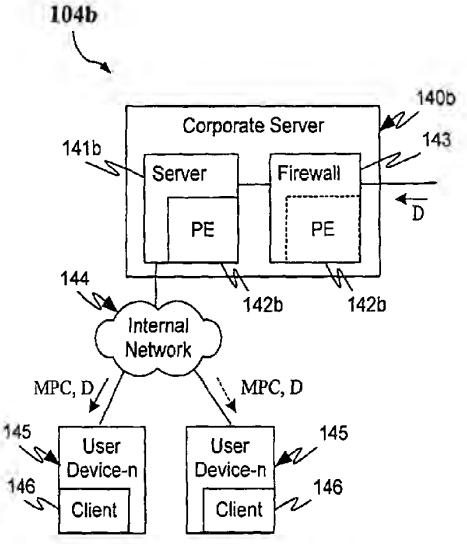


FIG. 1c

Case 3:17-cv-00183-BAS-NLS Document 1-5 Filed 07/01/16 PageID.151 Page 17 of 36

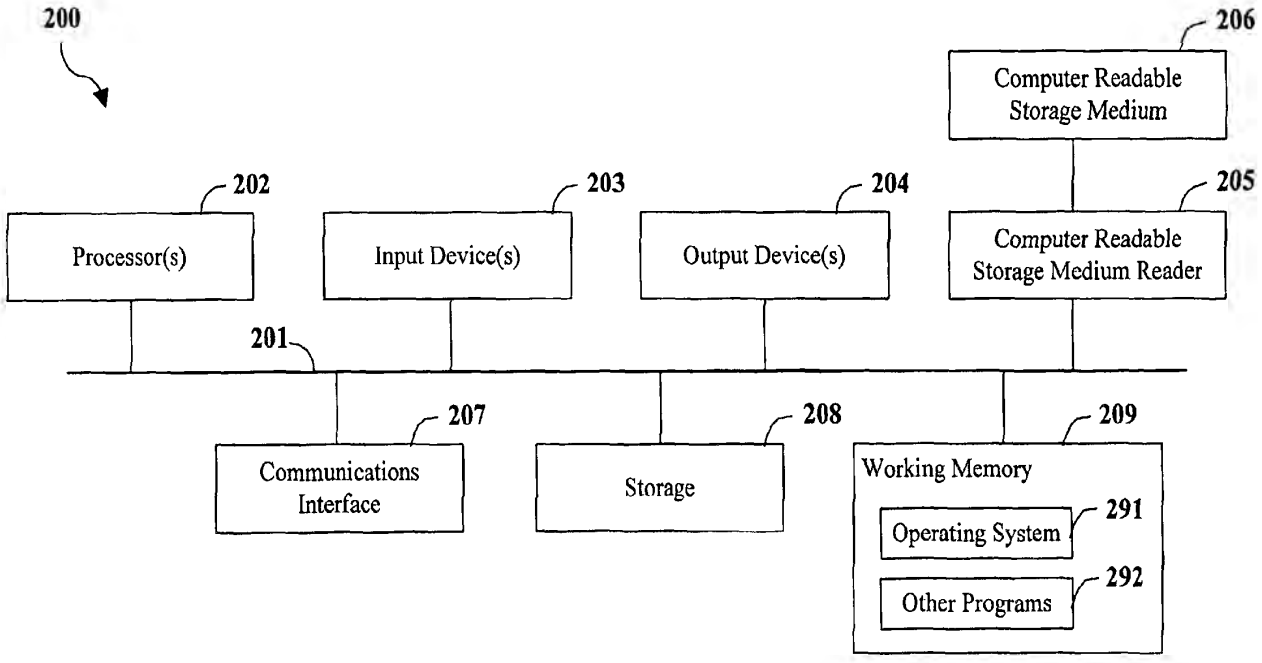


FIG. 2

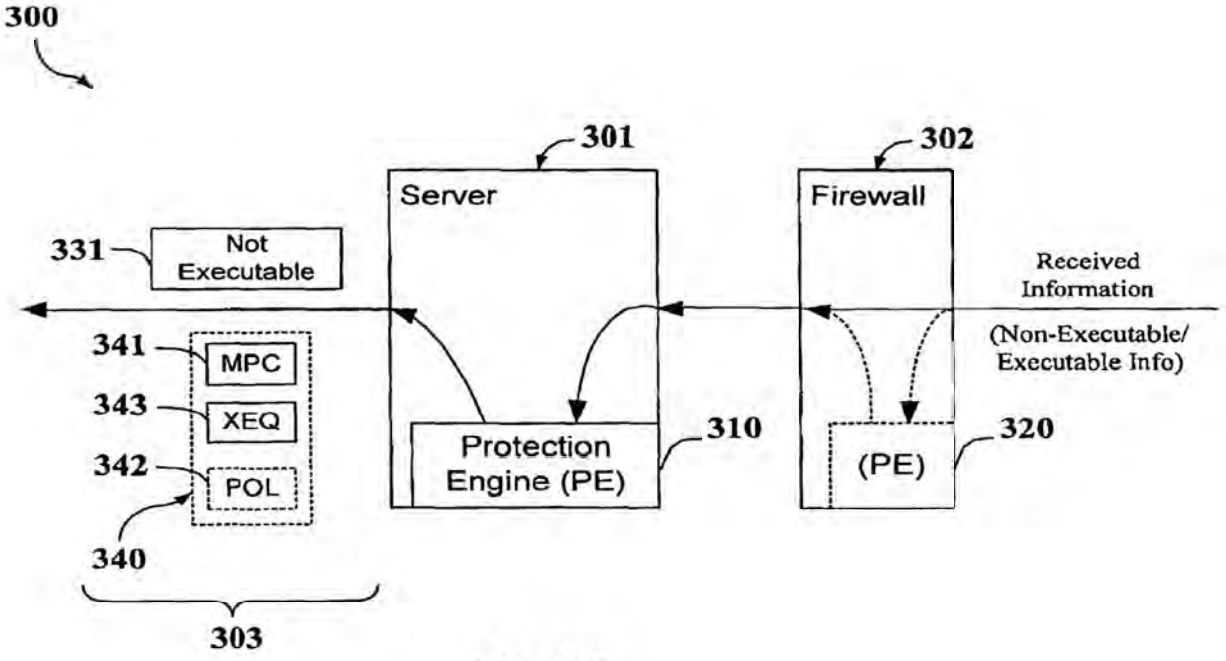

300
301
Server
302
Firewall
331
Not Executable
Received Information
(Non-Executable/ Executable Info)
341
MPC
343
XEQ
342
POL
340
303
Protection Engine (PE)
310
(PE)
320

FIG. 3

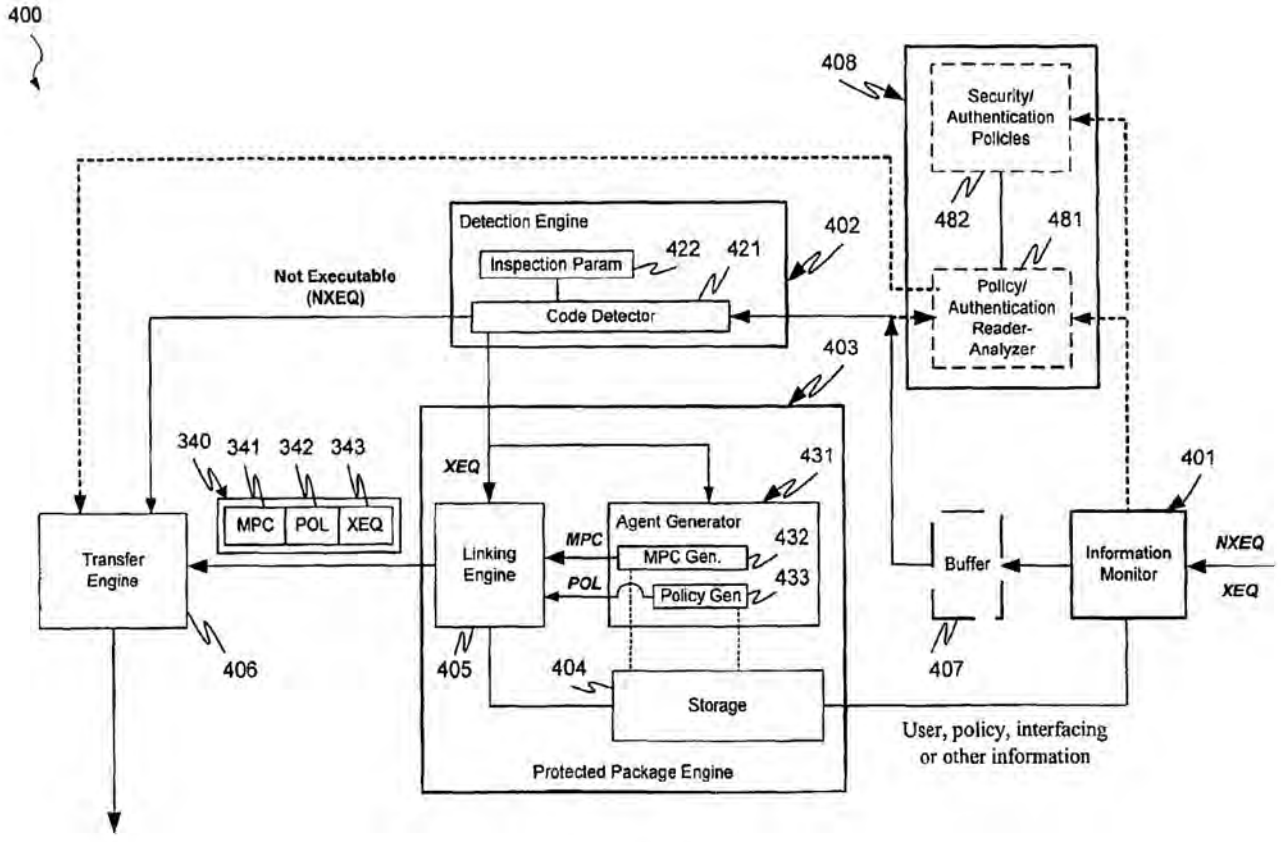

400
408
Security/ Authentication Policies
482
481
Policy/ Authentication Reader-Analyzer
Detection Engine
402
Inspection Param
422
421
Code Detector
Not Executable (NXEQ)
403
340
341
342
343
MPC
POL
XEQ
XEQ
431
Agent Generator
MPC
MPC Gen.
432
POL
Policy Gen
433
Linking Engine
Transfer Engine
406
405
404
Storage
Protected Package Engine
Buffer
407
401
Information Monitor
NXEQ
XEQ
User, policy, interfacing or other information

FIG. 4

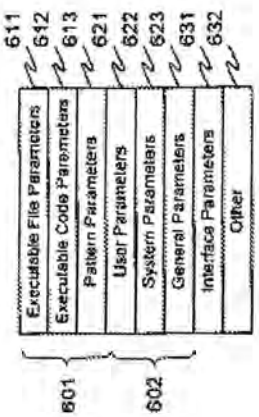


FIG. 6a

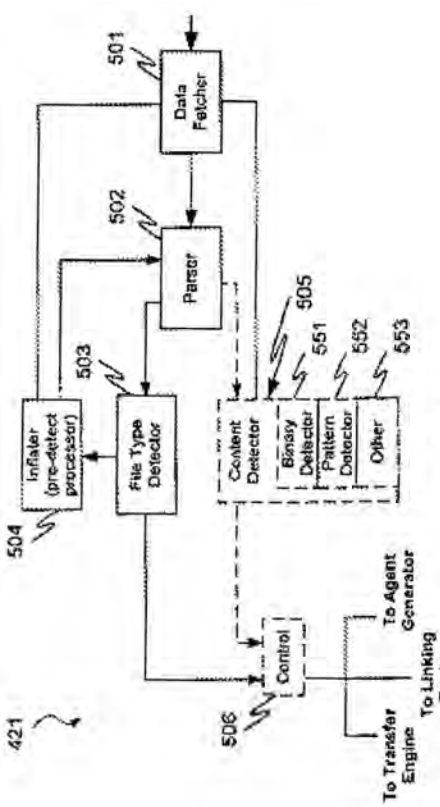


FIG. 5

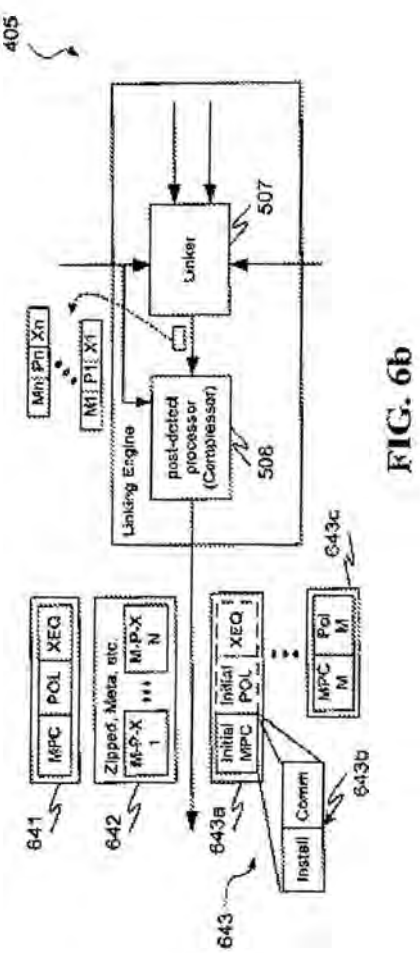


FIG. 6b

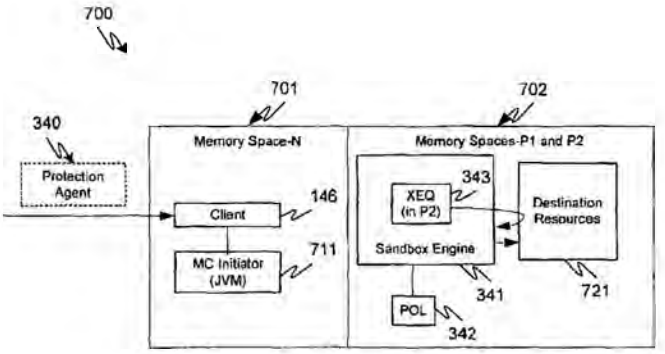


FIG. 7a

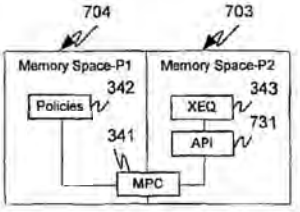


FIG. 7b

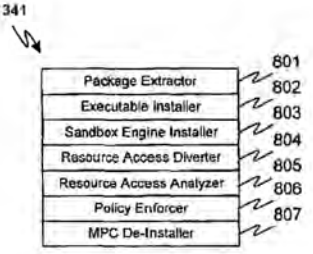


FIG. 8

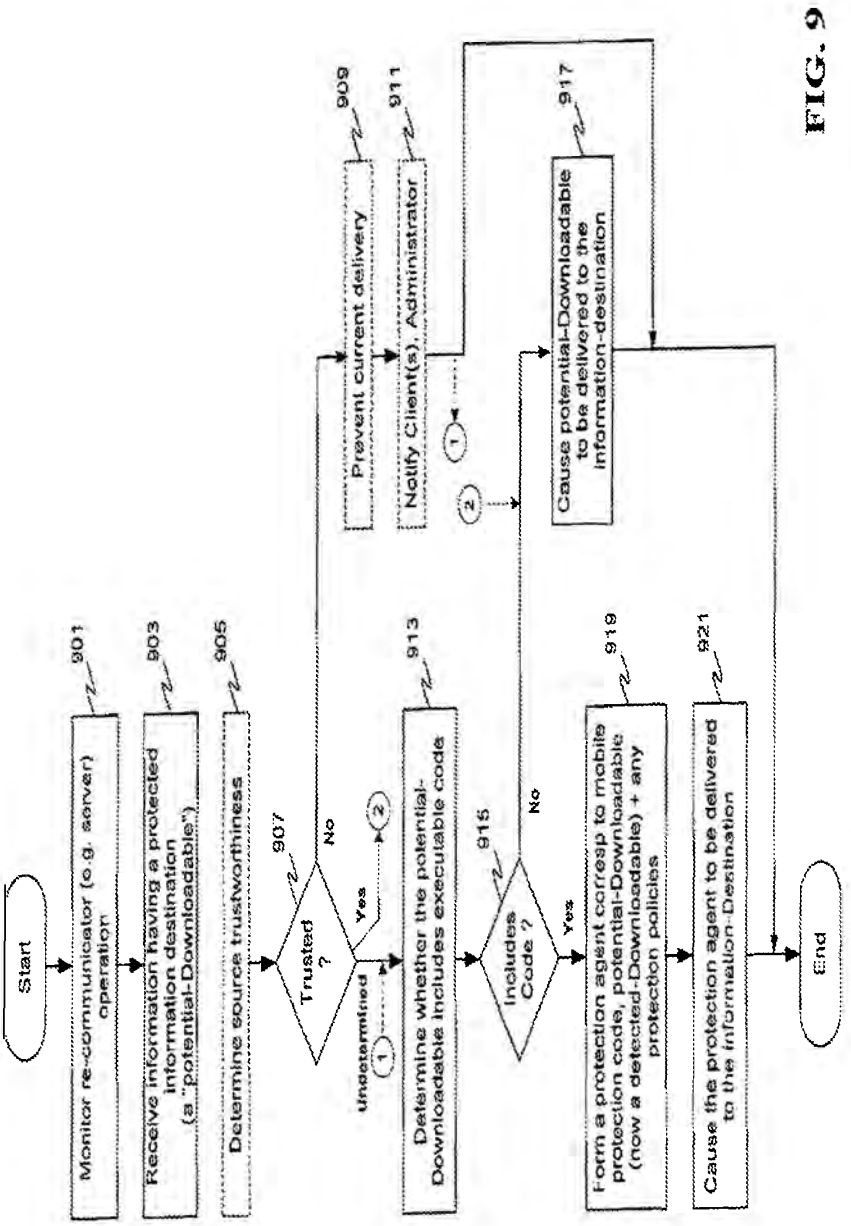

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
Trusted ?
907
No
Yes
Untrustworthy
Prevent current delivery
909
Notify Client(s), Administrator
911
Determine whether the potential-Downloadable includes executable code
913
Includes Code ?
915
No
Yes
Cause potential-Downloadable to be delivered to the information-destination
917
Form a protection agent corresp to mobile protection code, potential-Downloadable (now a detected-Downloadable) + any protection policies
919
Cause the protection agent to be delivered to the information-Destination
921
End

FIG. 9

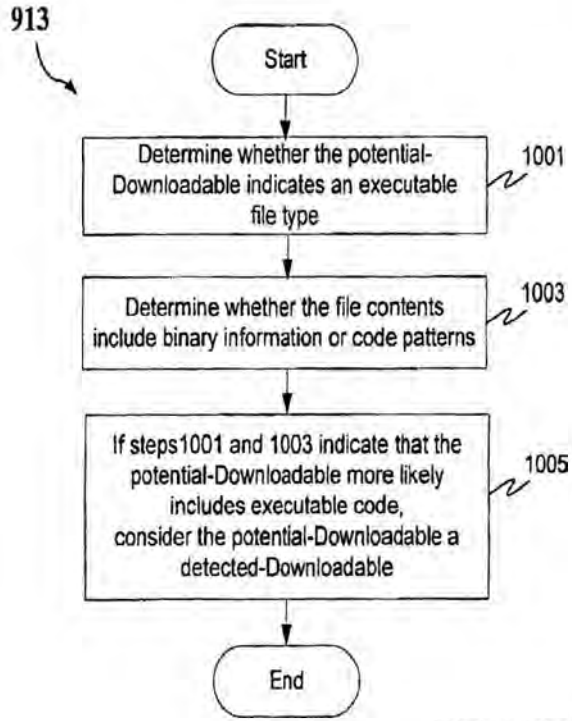


FIG. 10A

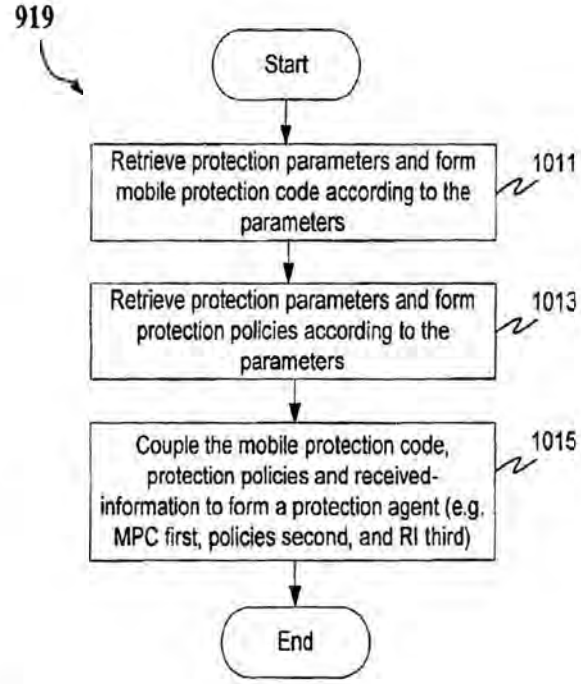


FIG. 10B

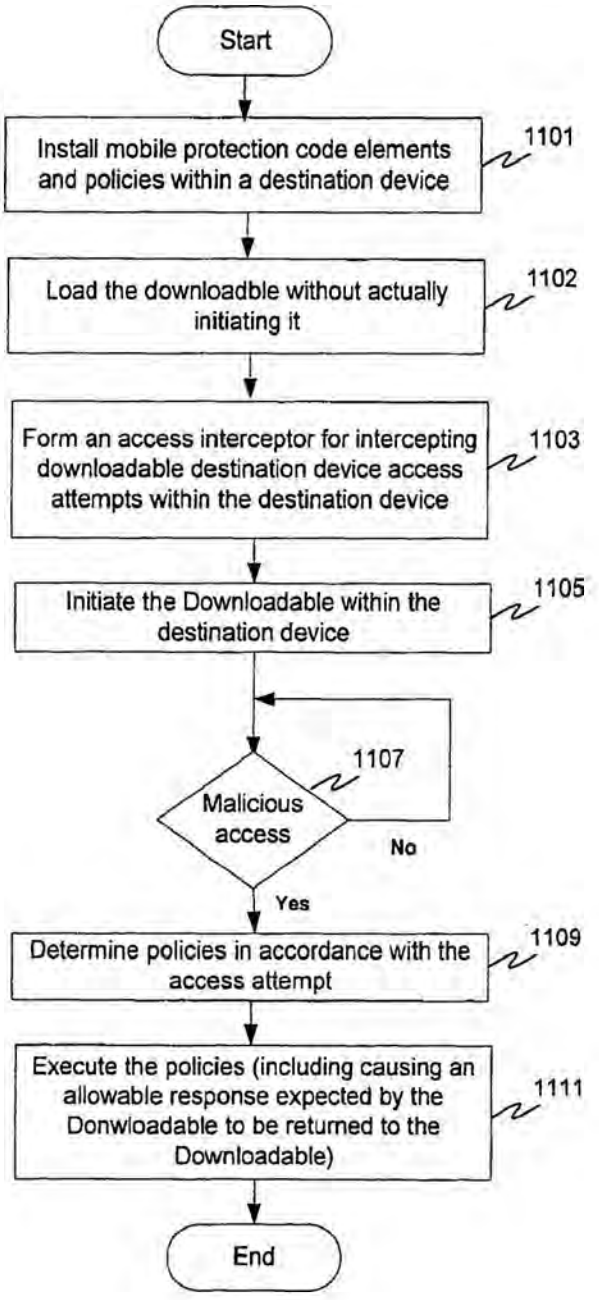

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable within the destination device
1105
1107
Malicious access
No
Yes
Determine policies in accordance with the access attempt
1109
Execute the policies (including causing an allowable response expected by the Donwloadable to be returned to the Downloadable)
1111
End

FIG. 11

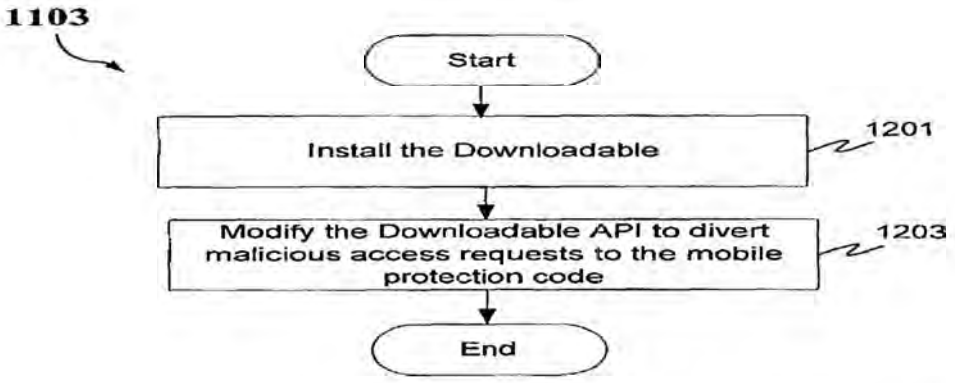


FIG. 12a

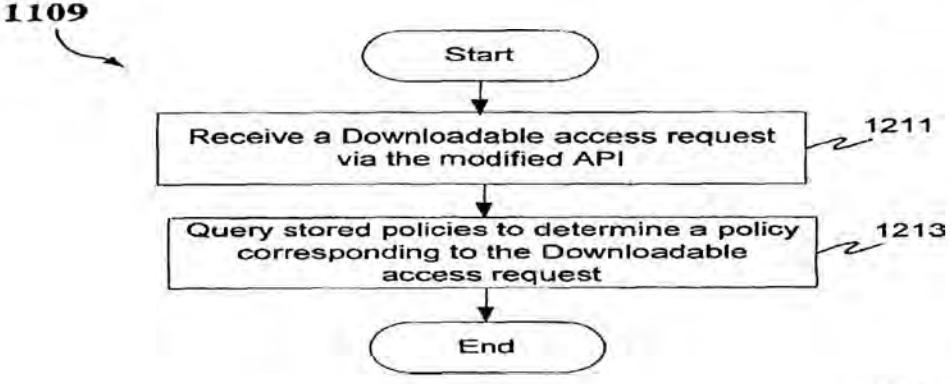


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application is a continuation of U.S. patent application Ser. No. 14/155,835, filed Jan. 15, 2014 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application Ser. No. 13/290,708, filed Nov. 7, 2011 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application Ser. No. 12/471,942, filed May 26, 2009 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 8,079,086, entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of assignee's U.S. patent application Ser. No. 11/370,114, filed Mar. 7, 2006 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 7,613,926, entitled "Method and System for Protecting a Computer and a Network from Hostile Downloadables," which is a continuation of assignee's U.S. patent application Ser. No. 09/861,229, filed on May 17, 2001 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 7,058,822, entitled "Malicious Mobile Code Runtime Monitoring System And Methods," all of which are hereby incorporated by reference. U.S. patent application Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, claims benefit of provisional U.S. patent 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. patent application Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, is also a Continuation-In-Part of assignee's 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 U.S. patent application Ser. No. 08/964,388, filed on Nov. 6, 1997 by inventor Shlomo Touboul, 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, which application claims the benefit of provisional U.S. application Ser. No. 60/030,639, filed Nov. 8, 1996 by inventors Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables." U.S. Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, is also a Continuation-In-Part of assignee's 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, filed Jan. 29, 1997 by inventor Shlomo Touboul, now U.S. Pat. No. 6,167,520, entitled "System and Method For Protecting a Client From Hostile Downloadables" which claims the benefit of U.S. provisional application No. 60/030,639, filed on Nov. 8, 1996 by inventor Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables."

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

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

FIG. **1***c* 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. **6***a* is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. **6***b* 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. **7***a* is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. **7***b* is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. **7***a*, 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. **10***a* 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. **10***b* 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. **12***a* is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. **12***b* 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. **1***a* through **1***c* illustrate a computer network system **100** according to an embodiment of the invention. FIG. **1***a* broadly illustrates system **100**, while FIGS. **1***b* and **12** of **1***c* illustrate exemplary protectable subsystem implementations corresponding with system **104** or **106** of FIG. **1***a*.

Beginning with FIG. **1***a*, 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. **1***a* 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. **1***b* and **1***c* 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** band firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements **11**) 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, 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-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, OSL, infrared or other suitable transceiver, etc. for providing inter-device commu-

nication 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 **110** 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 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 NM 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 (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 me 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 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 **20** (e.g. see FIG. **6***b*). 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 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. (Anyone 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**, inflator **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 as 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 inflator **504**. File inflator **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, inflator **504** returns such information to parser **502**. File inflator **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. **6***b*). 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 **643***b* of FIG. **6***b*. (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 (**643***a* of FIG. **6***b*). 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 **643***c*, 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. **6***a* 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. **6***b* 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 concatination 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. **7***a*, **7***b* and **8** illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. **7***a*, 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, PI. 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 IA T can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IA T 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.

Turning to FIG. **8** with reference to FIG. **7***b*, 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 deinstaller **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 PI) 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 as/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 **20** 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. **10***a* 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. **10***a*, 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. **10***b* 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. **12***a* example, the MPC can provide for intercepting Downloadable access attempts by a Down-

loadable 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 ill, 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. **12***b* 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 system for determining whether a downloadable is suspicious, comprising:

at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:

a plurality of operating system probes operating substantially in parallel for monitoring a plurality of subsystems of the operating system during runtime for an event caused from a request made by a downloadable;

an interrupter for interrupting processing of the request;

a first comparator coupled to the plurality of operating system probes for comparing information pertaining to the downloadable against a predetermined security policy, wherein the information pertaining to the downloadable includes information pertaining to an operation of the downloadable and distinct from information pertaining to the request; and

a response engine for performing a predetermined responsive action based on the comparison.

2. The system of claim **1**, wherein the plurality of operating system interfaces operating substantially in parallel for monitoring the operating system includes means for monitoring a request sent to a downloadable engine.

3. The system of claim **1**, further comprising means for determining whether information pertaining to the downloadable violates a security rule.

4. The system of claim **3**, further comprising means for determining whether violation of the security rule violates the security policy.

5. The system of claim **1**, further comprising:

a comparator for comparing information pertaining to the downloadable with information pertaining to a predetermined suspicious downloadable; and

a response engine for performing a predetermined responsive action based on the comparison with the information pertaining to the predetermined suspicious downloadable.

6. The system of claim **1**, wherein the predetermined responsive action includes storing results of the comparison in an event log.

7. The system of claim **1**, wherein the predetermined responsive action includes informing the user when the security policy has been violated.

8. The system of claim **1**, wherein the predetermined responsive action includes storing information on the Downloadable in a suspicious Downloadable database.

9. The system of claim **1**, wherein the predetermined responsive action includes discarding the Downloadable.

10. A system for reviewing an operating system call issued by a downloadable, comprising:

at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:

a plurality of operating system probes for monitoring substantially in parallel a plurality of subsystems of an operating system during runtime for an event caused from a request made by a Downloadable, wherein the plurality of subsystems includes a network system;

an interrupter for interrupting processing of the request;

a comparator coupled to the plurality of operating system probes for comparing information pertaining to the Downloadable against a predetermined security policy; and

a response engine for performing a predetermined responsive action based on the comparison.

11. The system of claim **10**, wherein the predetermined responsive action includes storing results of the comparison in an event log.

12. The system of claim **10**, wherein the predetermined responsive action includes informing the user when the security policy has been violated.

13. The system of claim **10**, wherein the predetermined responsive action includes storing information on the Downloadable in a suspicious Downloadable database.

14. The system of claim **10**, wherein the predetermined responsive action includes discarding the Downloadable.

15. The system of claim **10**, wherein the plurality of operating system probes operating substantially in parallel for monitoring the operating system includes means for monitoring a request sent to a downloadable engine.

16. The system of claim **10**, further comprising means for determining whether information pertaining to the downloadable violates a security rule.

* * * * *

Exhibit F

US009219755B2

(12) **United States Patent**
Touboul

(10) **Patent No.: US 9,219,755 B2**
(45) **Date of Patent: *Dec. 22, 2015**

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

(71) Applicant: **Finjan, Inc.**, East Palo Alto, CA (US)
(72) Inventor: **Shlomo Touboul**, Kefar-Haim (IL)
(73) Assignee: **Finjan, Inc.**, East Palo Alto, CA (US)

(*) 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.: **14/732,188**
(22) Filed: **Jun. 5, 2015**

(65) **Prior Publication Data**

US 2015/0288720 A1 Oct. 8, 2015

Related U.S. Application Data

(60) Division of application No. 14/619,363, filed on Feb. 11, 2015, which is a continuation of application No. 14/155,835, filed on Jan. 15, 2014, which is a continuation of application No. 13/290,708, filed on

(Continued)

(51) **Int. Cl.**
H04L 29/06 (2006.01)
G06F 21/51 (2013.01)
(Continued)

(52) **U.S. Cl.**
CPC ***H04L 63/20*** (2013.01); ***G06F 21/51*** (2013.01); ***G06F 21/53*** (2013.01); ***G06F 21/56*** (2013.01);
(Continued)

(58) **Field of Classification Search**
CPC ... H04L 63/20; H04L 63/1441; H04L 63/145; H04L 63/168; G06F 21/53; G06F 21/56; G06F 21/51; G06F 2221/033
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,562,305 A 12/1985 Gaffney, Jr.
4,864,616 A 9/1989 Pond et al.
(Continued)

FOREIGN PATENT DOCUMENTS

EP 0636977 7/1994
EP 0813133 12/1997
(Continued)

OTHER PUBLICATIONS

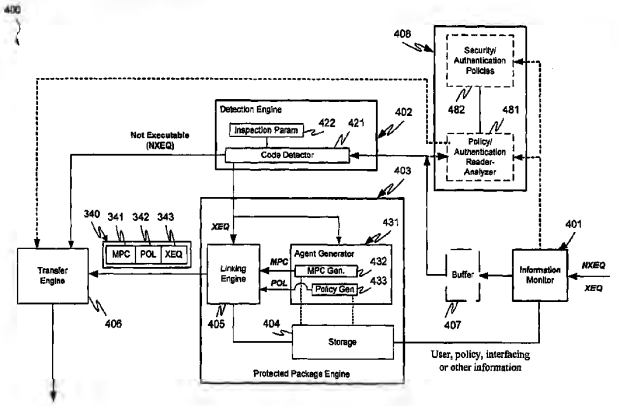
U.S. Appl. No. 60/174,952, filed Jan. 7, 2000, Lieno, et al.
(Continued)

Primary Examiner — Andrew Nalven
Assistant Examiner — Mary Li
(74) *Attorney, Agent, or Firm* — Dawn-Marie Bey; Bey & Cotropia PLLC

(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 TN 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 for monitoring information received, 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. 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.

8 Claims, 10 Drawing Sheets



Related U.S. Application Data

Nov. 7, 2011, now Pat. No. 8,677,494, which is a continuation of application No. 12/471,942, filed on May 26, 2009, now Pat. No. 8,079,086, which is a continuation of application No. 11/370,114, filed on Mar. 7, 2006, now Pat. No. 7,613,926, which is a continuation of application No. 09/861,229, filed on May 17, 2001, now Pat. No. 7,058,822, which is 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, and application No. 09/861,229, May 17, 2001, 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.

(60) Provisional application No. 60/030,639, filed on Nov. 8, 1996, provisional application No. 60/205,591, filed on May 17, 2000.

(51) **Int. Cl.**
G06F 21/56 (2013.01)
G06F 21/53 (2013.01)

(52) **U.S. Cl.**
CPC ***H04L 63/145*** (2013.01); ***H04L 63/1441*** (2013.01); ***H04L 63/168*** (2013.01); *G06F 2221/033* (2013.01)

(56) **References Cited**

U.S. PATENT DOCUMENTS

4,978,484	A	12/1990	Takashige et al.	264/40.1
5,050,212	A	9/1991	Dyson	380/25
5,077,677	A	12/1991	Murphy et al.	
5,263,147	A	11/1993	Francisco et al.	
5,278,901	A	1/1994	Shieh et al.	
5,283,830	A	2/1994	Hinsley et al.	
5,311,591	A	5/1994	Fischer	
5,313,616	A	5/1994	Cline et al.	395/500
5,319,776	A	6/1994	Rile et al.	
5,337,360	A	8/1994	Fischer	
5,345,595	A	9/1994	Johnson et al.	
5,359,659	A	10/1994	Rosenthal	
5,361,359	A	11/1994	Tajalli et al.	
5,389,196	A	2/1995	Bloomstein et al.	
5,398,196	A	3/1995	Chambers	
5,412,717	A	5/1995	Fischer	
5,414,833	A	5/1995	Hershey et al.	
5,421,006	A	5/1995	Jablon et al.	395/575
5,440,723	A	8/1995	Arnold et al.	
5,452,442	A	9/1995	Kephart	
5,471,614	A	11/1995	Kakimoto	
5,475,753	A	12/1995	Barbara et al.	
5,483,649	A	1/1996	Kuznetsov et al.	
5,485,409	A	1/1996	Gupta et al.	
5,485,575	A	1/1996	Chess et al.	
5,524,238	A	6/1996	Miller et al.	
5,572,643	A	11/1996	Judson	
5,579,509	A	11/1996	Furtney et al.	
5,606,668	A	2/1997	Shwed	
5,621,889	A	4/1997	Lermuzeaux et al.	
5,623,600	A	4/1997	Ji et al.	
5,623,601	A	4/1997	Vu	
5,625,693	A	4/1997	Rohatgi et al.	380/23
5,638,446	A	6/1997	Rubin	
5,666,411	A	9/1997	McCarty	
5,673,316	A	9/1997	Auerbach et al.	
5,675,711	A	10/1997	Kephart et al.	
5,678,041	A	10/1997	Baker et al.	
5,680,461	A	10/1997	McManis	
5,692,047	A	11/1997	McManis	
5,692,124	A	11/1997	Holden et al.	
5,696,822	A	12/1997	Nachenberg	
5,699,512	A	12/1997	Moriyasu et al.	
5,720,033	A	2/1998	Ceo	
5,724,425	A	3/1998	Chang et al.	
5,740,248	A	4/1998	Fieres et al.	
5,740,441	A	4/1998	Yellin et al.	
5,748,960	A	5/1998	Fischer	395/683
5,757,915	A	5/1998	Aucsmith et al.	
5,761,421	A	6/1998	Van Hoff et al.	
5,765,030	A	6/1998	Nachenberg et al.	
5,765,205	A	6/1998	Breslau et al.	
5,784,459	A	7/1998	Devarakonda et al.	
5,787,175	A	7/1998	Carter	
5,796,952	A	8/1998	Davis et al.	
5,805,829	A *	9/1998	Cohen et al.	709/202
5,809,230	A	9/1998	Pereira	
5,815,709	A	9/1998	Waldo et al.	
5,825,877	A	10/1998	Dan et al.	
5,832,208	A	11/1998	Chen et al.	
5,832,274	A	11/1998	Cutler et al.	
5,842,002	A	11/1998	Schnurer et al.	
5,842,040	A	11/1998	Hughes et al.	
5,845,281	A	12/1998	Benson et al.	
5,850,559	A	12/1998	Angelo et al.	
5,854,916	A	12/1998	Nachenberg	
5,859,966	A	1/1999	Hayman et al.	
5,860,011	A	1/1999	Kolawa et al.	
5,864,683	A	1/1999	Boebert et al.	
5,867,651	A	2/1999	Dan et al.	
5,878,258	A	3/1999	Pizi et al.	
5,881,151	A	3/1999	Yamamoto	
5,884,003	A	3/1999	Cloud et al.	385/135
5,884,033	A	3/1999	Duvall et al.	
5,889,943	A	3/1999	Ji et al.	
5,889,952	A	3/1999	Hunnicutt et al.	
5,892,904	A	4/1999	Atkinson et al.	
5,894,516	A	4/1999	Brandenburg	
5,911,043	A	6/1999	Duffy et al.	
5,933,503	A	8/1999	Schell et al.	
5,940,590	A	8/1999	Lynne et al.	
5,944,821	A	8/1999	Angelo	713/200
5,949,415	A	9/1999	Lin et al.	
5,951,698	A	9/1999	Chen et al.	
5,956,481	A	9/1999	Walsh et al.	
5,958,050	A	9/1999	Griffin et al.	
5,960,170	A	9/1999	Chen et al.	
5,963,742	A	10/1999	Williams	
5,964,889	A	10/1999	Nachenberg	
5,974,549	A	10/1999	Golan	
5,978,484	A	11/1999	Apperson et al.	
5,978,840	A	11/1999	Nguyen et al.	
5,983,267	A	11/1999	Shklar et al.	
5,983,348	A	11/1999	Ji	
5,987,123	A	11/1999	Scott et al.	
5,987,610	A	11/1999	Franczek et al.	
5,987,611	A	11/1999	Freund	
5,991,810	A	11/1999	Shapiro et al.	
6,021,510	A	2/2000	Nachenberg	
6,049,671	A	4/2000	Slivka et al.	395/712
6,058,482	A	5/2000	Liu	
6,065,118	A	5/2000	Bull et al.	
6,067,575	A	5/2000	McManis et al.	
6,070,239	A	5/2000	McManis	
6,073,175	A	6/2000	Tavs et al.	
6,083,279	A	7/2000	Cuomo et al.	
6,088,801	A	7/2000	Grecsek	
6,088,803	A	7/2000	Tso et al.	
6,092,147	A	7/2000	Levy et al.	
6,092,194	A	7/2000	Touboul	
6,125,390	A	9/2000	Touboul	
6,128,774	A	10/2000	Necula et al.	
6,148,083	A	11/2000	Fieres et al.	380/255
6,148,401	A	11/2000	Devanbu et al.	
6,154,842	A	11/2000	Van Peursem	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
6,167,522	A	12/2000	Lee et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,170,012	B1	1/2001	Coss et al.	
6,170,057	B1	1/2001	Inoue et al.	
6,175,924	B1	1/2001	Arnold	
6,192,512	B1	2/2001	Chess	
6,205,551	B1	3/2001	Grosse	
6,230,288	B1	5/2001	Kuo et al.	
6,253,323	B1	6/2001	Cox et al.	
6,253,370	B1	6/2001	Abadi et al.	
6,263,442	B1	7/2001	Mueller et al.	
6,266,664	B1	7/2001	Russell-Falla et al.	
6,266,773	B1	7/2001	Kisor et al.	
6,272,641	B1	8/2001	Ji	
6,275,937	B1	8/2001	Hailpern et al.	
6,275,938	B1	8/2001	Bond et al.	
6,317,868	B1	11/2001	Grimm et al.	
6,321,334	B1	11/2001	Jerger et al.	
6,324,685	B1	11/2001	Balassanian	
6,336,140	B1	1/2002	Elgressy et al.	
6,339,829	B1	1/2002	Beadle et al.	
6,343,376	B1	1/2002	Saxe et al.	
6,351,816	B1	2/2002	Mueller et al.	
6,357,008	B1	3/2002	Nachenberg	
6,367,012	B1	4/2002	Atkinson et al.	
6,401,241	B1	6/2002	Taylor	
6,418,444	B1	7/2002	Raduchel et al.	
6,425,058	B1	7/2002	Arimilli et al.	
6,434,668	B1	8/2002	Arimilli et al.	
6,434,669	B1	8/2002	Arimilli et al.	
6,449,723	B1	9/2002	Elgressy et al.	
6,470,450	B1	10/2002	Langford et al.	
6,473,800	B1	10/2002	Jerger et al.	
6,480,962	B1	11/2002	Touboul	
6,487,666	B1	11/2002	Shanklin et al.	
6,505,300	B2	1/2003	Chan et al.	
6,519,679	B2	2/2003	Devireddy et al.	
6,526,565	B1	2/2003	Nally	
6,560,774	B1	5/2003	Gordon et al.	
6,567,917	B1	5/2003	Ziese	
6,571,338	B1	5/2003	Shaio et al.	
6,594,783	B1	7/2003	Dollin et al.	
6,598,033	B2	7/2003	Ross et al.	
6,604,143	B1	8/2003	Nagar et al.	
6,622,247	B1	9/2003	Isaak	
6,622,781	B2	9/2003	Chou	
6,640,284	B1	10/2003	Shaw et al.	
6,643,696	B2	11/2003	Davis et al.	
6,668,325	B1	12/2003	Collberg et al.	
6,691,230	B1	2/2004	Bardon	
6,697,950	B1	2/2004	Ko	
6,704,798	B1	3/2004	Mogul	
6,721,721	B1	4/2004	Bates et al.	
6,725,424	B1	4/2004	Schwerdtfeger et al.	
6,732,179	B1	5/2004	Brown et al.	
6,748,538	B1	6/2004	Chan et al.	
6,772,214	B1	8/2004	McClain et al.	
6,772,225	B1	8/2004	Jennings, III et al.	
6,772,332	B1	8/2004	Boebert et al.	
6,775,536	B1	8/2004	Geiger et al.	455/411
6,804,780	B1	10/2004	Touboul	
6,829,746	B1	12/2004	Schwerdtfeger et al.	
6,832,222	B1	12/2004	Zimowski	
6,842,861	B1	1/2005	Cox et al.	
6,886,013	B1	4/2005	Beranek	
6,901,519	B1	5/2005	Stewart et al.	
6,917,953	B2	7/2005	Simon et al.	
6,917,960	B1	7/2005	Decasper et al.	
6,944,772	B2	9/2005	Dozortsev	
6,981,138	B2	12/2005	Douceur et al.	
6,996,841	B2	2/2006	Kadyk et al.	
6,996,845	B1	2/2006	Hurst et al.	726/25
7,039,946	B1	5/2006	Binding	
7,047,369	B1	5/2006	Gruper et al.	
7,051,164	B2	5/2006	Smith	
7,052,822	B2	5/2006	Goto	430/283.1
7,058,822	B2	6/2006	Edery et al.	
7,076,650	B1	7/2006	Sonnenberg	
7,089,591	B1	8/2006	Chi	
7,093,135	B1	8/2006	Radatti et al.	
7,096,496	B1	8/2006	Challener et al.	
7,103,714	B1	9/2006	Jacobs et al.	
7,143,444	B2	11/2006	Porras et al.	
7,181,725	B1	2/2007	Posegga et al.	
7,185,210	B1 *	2/2007	Faden	726/2
7,210,041	B1	4/2007	Gryaznov et al.	
7,243,366	B2	7/2007	Medvinsky et al.	
7,308,648	B1	12/2007	Buchthal et al.	
7,321,970	B2	1/2008	Watkins et al.	
7,343,604	B2	3/2008	Grabamik et al.	
7,350,204	B2	3/2008	Lambert et al.	
7,363,361	B2	4/2008	Tewari et al.	
7,363,493	B2	4/2008	Dotan	
7,398,553	B1	7/2008	Li	
7,409,718	B1	8/2008	Hong et al.	726/24
7,418,731	B2	8/2008	Touboul	
7,418,732	B2	8/2008	Campbell et al.	
7,434,260	B2	10/2008	Hong et al.	
7,461,262	B1	12/2008	O'Toole, Jr.	
7,467,304	B2	12/2008	Bar-El et al.	
7,487,540	B2	2/2009	Shipp	
7,493,658	B2	2/2009	Shipp	
7,506,155	B1	3/2009	Stewart et al.	
7,567,800	B2	7/2009	Uematsu et al.	
7,568,114	B1	7/2009	Schlafly	
7,574,738	B2	8/2009	Daude et al.	
7,587,755	B2	9/2009	Kramer	726/4
7,613,918	B2	11/2009	Ben-Itzhak	713/150
7,613,926	B2	11/2009	Edery et al.	
7,620,733	B1	11/2009	Tzakikario et al.	
7,627,896	B2	12/2009	Herrmann	
7,636,851	B2	12/2009	Hamblin et al.	
7,636,945	B2	12/2009	Chandnani et al.	
7,640,427	B2	12/2009	Callas et al.	
7,647,633	B2	1/2010	Edery et al.	
7,660,902	B2	2/2010	Graham et al.	
7,694,135	B2	4/2010	Rowan et al.	
7,694,323	B2	4/2010	LaJoie et al.	
7,694,328	B2	4/2010	Joshi et al.	
7,698,549	B2	4/2010	Thornton et al.	
7,735,138	B2	6/2010	Zhao	
7,752,242	B1	7/2010	Satish et al.	707/826
7,756,996	B2	7/2010	Touboul	
7,757,289	B2	7/2010	Gruzman et al.	
7,761,594	B1	7/2010	Mowat	
7,761,605	B1	7/2010	Rothwell et al.	
7,774,619	B2	8/2010	Paaske et al.	
7,797,727	B1	9/2010	Miller et al.	726/1
7,805,758	B2	9/2010	Hasegawa et al.	
7,822,974	B2	10/2010	Kinser et al.	
7,849,306	B2	12/2010	Takeshima et al.	
7,865,961	B2	1/2011	Hasegawa et al.	
7,930,738	B1	4/2011	Petersen	726/16
7,975,305	B2	7/2011	Rubin et al.	
7,984,304	B1	7/2011	Waldspurger et al.	
7,984,479	B2	7/2011	Brabson et al.	
8,015,182	B2	9/2011	Yuval et al.	
8,078,740	B2	12/2011	Franco et al.	
8,079,086	B1	12/2011	Edery et al.	
8,126,866	B1	2/2012	Barton et al.	
8,141,154	B2	3/2012	Gruzman et al.	726/22
8,225,392	B2	7/2012	Dubrovsky	
8,225,408	B2	7/2012	Rubin et al.	
8,244,910	B2	8/2012	Davis	
8,281,401	B2	10/2012	Pennington et al.	
8,321,662	B2	11/2012	Hazlewood et al.	
8,347,392	B2	1/2013	Chess et al.	
8,364,951	B2	1/2013	Peterka et al.	
8,374,958	B2	2/2013	Blott et al.	
8,375,423	B2	2/2013	Hays et al.	726/5
8,402,012	B1	3/2013	Herz et al.	
8,424,090	B2	4/2013	Kang et al.	726/24
8,438,499	B2	5/2013	Dixon et al.	
8,522,350	B2	8/2013	Davenport et al.	726/24
8,677,494	B2	3/2014	Edery et al.	
2001/0025311	A1	9/2001	Arai et al.	

(56) **References Cited**

U.S. PATENT DOCUMENTS

2002/0026511	A1	2/2002	Garcia-Luna-Aceves et al.	
2002/0059144	A1	5/2002	Meffert et al.	
2002/0059157	A1	5/2002	Spooner et al.	
2002/0065775	A1	5/2002	Monaghan	
2002/0065776	A1	5/2002	Calder et al.	
2002/0066024	A1	5/2002	Schmall et al.	
2002/0073330	A1	6/2002	Chandnani et al.	
2002/0099829	A1	7/2002	Richards et al.	
2002/0165912	A1	11/2002	Wenocur et al.	
2002/0198953	A1	12/2002	O'Rourke et al.	
2003/0009571	A1	1/2003	Bavadekar	
2003/0014662	A1	1/2003	Gupta et al.	
2003/0028616	A1	2/2003	Aoki et al.	
2003/0046412	A1	3/2003	Tsunoda et al.	
2003/0063752	A1	4/2003	Medvinsky et al.	
2003/0074190	A1	4/2003	Allison	
2003/0097582	A1	5/2003	Audebert et al.	
2003/0097591	A1	5/2003	Pham et al.	
2003/0101358	A1	5/2003	Porras et al.	
2003/0154265	A1	8/2003	Raffaele et al.	
2003/0172291	A1	9/2003	Judge et al.	
2003/0191801	A1	10/2003	Paul	
2003/0208490	A1	11/2003	Larrea et al.	
2003/0225883	A1	12/2003	Greaves et al.	
2004/0006706	A1	1/2004	Erlingsson	713/200
2004/0015538	A1	1/2004	Agarwalla et al.	
2004/0015723	A1	1/2004	Pham et al.	
2004/0016948	A1	1/2004	Lin	
2004/0039924	A1	2/2004	Baldwin et al.	
2004/0054854	A1	3/2004	Thiyagaranjan et al.	
2004/0073668	A1	4/2004	Bhat et al.	
2004/0073811	A1	4/2004	Sanin	
2004/0088425	A1	5/2004	Rubinstein et al.	
2004/0107363	A1	6/2004	Monteverde	
2004/0111413	A1	6/2004	Birbo et al.	
2004/0117426	A1	6/2004	Rudkin et al.	
2004/0117623	A1	6/2004	Kalogridis et al.	
2004/0122926	A1	6/2004	Moore et al.	
2004/0133520	A1	7/2004	Callas et al.	
2004/0148281	A1	7/2004	Bates et al.	
2004/0158741	A1	8/2004	Schneider	
2004/0230744	A1	11/2004	Teh	
2004/0243839	A1	12/2004	Bhatia et al.	
2004/0249965	A1	12/2004	Huggins et al.	
2005/0021791	A1	1/2005	Sakiyama et al.	709/229
2005/0050338	A1	3/2005	Lang et al.	
2005/0114653	A1	5/2005	Sudia	
2005/0172338	A1	8/2005	Sandu et al.	
2005/0193429	A1	9/2005	Demopoulos et al.	
2005/0246383	A1	11/2005	Desai et al.	
2006/0031207	A1	2/2006	Bjamestam et al.	
2006/0031352	A1	2/2006	Marston et al.	
2006/0048224	A1	3/2006	Duncan et al.	
2006/0143442	A1	6/2006	Smith	
2006/0236392	A1	10/2006	Thomas et al.	
2006/0294366	A1	12/2006	Nadalin et al.	
2007/0011739	A1	1/2007	Zamir et al.	
2007/0016770	A1	1/2007	Dumitru	
2007/0113282	A1	5/2007	Ross	
2007/0134082	A1	6/2007	Blomgren et al.	
2008/0066160	A1	3/2008	Becker et al.	
2008/0083012	A1	4/2008	Yu et al.	726/1
2009/0125612	A1	5/2009	Rabetge et al.	
2009/0193497	A1	7/2009	Kikuchi et al.	726/1
2010/0195909	A1	8/2010	Wasson et al.	
2011/0099621	A1	4/2011	Lizarraga et al.	
2012/0117651	A1	5/2012	Edery et al.	

FOREIGN PATENT DOCUMENTS

EP	0909072		4/1999
EP	0909075		4/1999
EP	1021276		7/2000
EP	1 058 199	A2	12/2000
EP	1091276		4/2001
EP	1119151		7/2001
EP	1132796		9/2001
IL	120420		12/1999
IL	129729		5/2005
JP	08-263447		10/1996
JP	H 1021134		1/1998
JP	01-256045	A	9/2001
JP	2002183019		6/2002
JP	02-358229		12/2002
JP	02-358253	A	12/2002
JP	2002373108		12/2002
JP	2003177991		6/2003
JP	3952315		5/2007
WO	WO 93/25024		12/1993
WO	WO 95/27249		10/1995
WO	WO 95/33237		12/1995
WO	WO 97/12322		4/1997
WO	WO 97/31698		9/1997
WO	WO 98/21683		5/1998
WO	WO 98/31124		7/1998
WO	WO 98/40993		9/1998
WO	WO 99/16225		4/1999
WO	WO 99/35583		7/1999
WO	WO 00/08793		2/2000
WO	WO 01/22682	A2	3/2001
WO	WO 01/25882		4/2001
WO	WO 01/88673		11/2001
WO	WO 01/98907		12/2001
WO	WO 01/99374		12/2001
WO	WO 02/13479		2/2002
WO	WO 2004/063948		7/2004
WO	WO 2006/025050		3/2006

OTHER PUBLICATIONS

U.S. Appl. No. 60/163,361, filed Nov. 3, 1999, Geiger, et al.
U.S. Appl. No. 60/156,872, filed Sep. 39, 1999, Huang, et al.
U.S Appl. No. 60/104,072, filed Oct. 13, 1998, Hicken.
U.S. Appl. No. 60/093,559, filed Jul. 21, 1998, Maloney, et al.
ThunderByte Antivirus for Windows.
InterScan Virus Wall from Trend Micro.
ViruSafe from Eliashim.
Intel LANProtect from Intel.
The Java Security Manager from Sun MicroSystems.
McAfee Web Shield.
McAfee WebScan.
McAfee VirusScan.
Dr. Solomon's Antivirus Toolkit for Windows 95.
Dr. Solomon's Antivirus Tookit for Windows NT.
Dr. Solomon's WinGuard.
Dr. Solomon's Virus Guard.
Dr. Solomon's Virus Shield.
Dr. Solomon's Virex.
Dr. Solomon's "Merlin" Anti-Virus Engine.
Dr. Solomon's/McAfee "Olympus" Anti-Virus Engine.
Norton AntiVirus™ for Windows® 95 User's Guide, Published by Symantec in 1995 (179 pp.).
Jaeger, et al., "Building Systems that Flexibly Control Downloadable Executable Content," Proceedings of the Sixth USENIX UNIX Security Symposium, Jul. 1996 (19 pp.).
Rasmusson, Andreas and Jansson, Sverker, "Personal Security Assistance for Secure Internet Commerce," Sep. 16, 1996 (12 pp.), Available as dvi, ps, html, at http://www.sics.se/~ara/papers/NSP96.html.
Bharat, et al., "Migratory Applications," Nov. 15, 1995 (10 pp.).
Dean, Drew, et al., "Java Security: From HotJava to Netscape and Beyond," 1996 IEEE Symposium on Security and Privacy, May 6, 1996 (11 pp.).
Sterbenz, Andreas, "An Evaluation of the Java Security Model," IEEE, Dec. 1996 (13 pp.).
Fritzinger, J. Steven, et al., Java Security, Sun Microsystems, Dec. 1996 (7 pp.).
Bank, Joseph A., "Java Security," Dec. 8, 1995 (14 pp.).
Chappell, "Understanding ActiveX and OLE: A Guide for Developers and Managers (Strategic Technology)," Sep. 1, 1996, Microsoft Press (91 pp.).

(56) **References Cited**

OTHER PUBLICATIONS

Crosbie, et al., "Active Defense of a Computer System Using Autonomous Agents," Feb. 15, 1995 (14 pp.).
"Trend Micro's Virus Protection Added to Sun Microsystems Netra Internet Servers," Business Wire, Oct. 1, 1996, available at http://www.cs.indiana.edu/~kinzler/pubs/viruswall.html.
"Symantec Announces Norton Antivirus 2.0 for Windows NT," Symantec Corporation press release, Sep. 16, 1996, available at http://www.symantec.com/about/news/release/article.jsp?prid=19960916_01.
"Dark Avenger Mutation Engine No Threat to Protected PCs," McAfee, Inc., press release, May 11, 1992, available at http://securitydigest.org/virus/mirror/www.phreak.orgvirus1/1992/vin105.191.
Gryaznov, D. O., "Scanners of the Year 2000: Heuristics," Proceedings of the Fifth International Virus Bulletin Conference, pp. 225-234 (1995), available at http://vxheavens.com/lib/adg00.html.
"Symantec Announces Norton Internet Email Gateway at Internet World—Booth # 369 on Dec. 11, 12, and 13," Symantec Corporation press release, Dec. 11, 1996, available at http://www.symantec.com/about/news/release/article.jsp?prid=19961211_03.
"Internet Security Gets Less Costly and Easier to Manage: Integralis Announces MIMEsweeper Compatible with Check Point FireWall-1 on Single NT Server; E-mail virus detection and content management can reside on Firewall server, saving money and supports costs" [online], The Free Library, Sep. 16, 1996 [retrieved on Mar. 20, 2012], 4 pp., Retrieved From the Internet: http://www.thefreelibrary.com/Internet+Security+gets+Less+Costly+and+Easier+to+Manage%3a+Integralis...a018675791%22%3Einternet%20Security%20gets%20Less%20Costly%20and%20Easier%20to%20Manage:%20Integralis%20Announces%20MIMEsweeper%20Compatible%20with%20Check%20Point%20FireWall-1%20on%20Single%20NT%20Server;%20E-mail%20virus%20detection%20and%20content%20management%20can%20reside%20on%20Firewall%20server,%20saving%20money%20and%20support%20costs (Current Link: http://www.thefreelibrary.com/Internet Security gets Less Costly and Easier to Manage: Integralis . . . -a018675791).
"The Java Language Specification," by Gosling, et al. (1996).
"The Java Virtual Machine Specification," by Tim Lindholm and Frank Yellin (1997).
"Java Security and a Firewall Extension for Authenticity Control of Java Applets," by Magnus Johansson (Jan. 29, 1997).
"Static Analysis of Programs With Application to Malicious Code Detection," by Raymond Lo, Dissertation, University of California, 193 pp. (1992).
File History for U.S. Pat. No. 6,804,780.
"Virus Detection Alternatives," by Patrick Min, http://vx.netlux.org/lib/apm00.html (Jul. 1992).
"Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns," by LeCharlier, et al. (Sep. 1995).
The Giant Black Book of Computer Viruses by Mark Ludwig (1995).
HotJava: The Security Story.
"The Java Filter," [retrieved on Jan. 2, 2011], 1 p., Retrieved from the Internet: http://www.cs.princeton.edu/sip/JavaFilter/index.html.
"Improved JavaScript and Java Screening Function," by Claunch (May 4, 1996), available at http://groups.google.com/group/muc.lists.firewalls/msg/96bc8043476ab0b7.
"New Version of Java, JavaScript, ActiveX Screening," by Claunch (Jul. 3, 1996), availableathttp://groups.google.com/group/muc.lists.firewalls/msg/41a45c2a9d5373f2.
"Identifying and Controlling Undesirable Program Behaviors," by Maria King, Proceedings of the 14th National Computer Security Conference, 1991.
"PACL's: An Access Control List Approach to Anti-Viral Security," by Wichers, et al., in USENIX Workshop Proceedings, UNIX Security II, pp. 71-82, 1990.
Endrijonas, Janet, "RX PC The Anti-Virus Handbook," Published in the U.S. in 1993 by TAB Books, a division of McGraw-Hill, Inc. (201 pp.).
"Secure Code Distribution," by X. Nick Zhang (Jun. 1997).
Omura, Jim K., "Novel Applications of Cryptography in Digital Communications," IEEE Communications Magazine, pp. 21-29, May 1990.
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/cgibin/iel_cgi?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts.
IBM AntiVirus User's Guide Version 2.4, International Business Machines Corporation, pp. 6-7, Nov. 15, 1995.
Leach, Norvin, et al., "IE 3.0 Applets Will Earn Certification," PC Week, vol. 13, No. 29, 2 pp., Jul. 22, 1996.
"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.
"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.
Microsoft® Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet," Microsoft Corporation, including Abstract, Contents, Introduction, and pp. 1-10, Oct. 1996.
"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.
Company Profile, "Finjan-Safe Surfing. The Java Security Solutions Provider," Article published on the Internet by Finjan Software Ltd., 3 pp., Oct. 31, 1996.
"Finjan Announces Major Power Boost and New Features for SurfinShield™ 2.0," Las Vegas Convention Center/Pavillion 5 P5551, 3 pp., Nov. 18, 1996.
"Java Security: Issues & Solutions," Article published on the Internet by Finjan Software Ltd., 10 pp., 1996.
"Products," Article published on the Internet, "Revolutionary Security for a New Computing Paradigm" regarding SurfinGate™ , 7 pp.
Mark LaDue, "Online Business Consultant: Java Security: Whose Business Is It?," Article published on the Internet, Home Page Press, Inc., 4 pp., 1996.
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/authcodee/signfaq.asp#9, pp. 1-13.
Zhang, X. N., "Secure Code Distribution," IEEE/IEE Electronic Library online, Computer, vol. 30, Issue 6, pp. 76-79, Jun. 1997.
Binstock, Andrew, "Multithreading, Hyper-Threading, Multiprocessing: Now, What's the Difference?," http://www.intel.com/cd/ids/developer/asmo-na/eng/20456.htm, Pacific Data Works, LLC, downloaded Jul. 7, 2008, 7 pp.
VirexPC Version 2.0 or later from Microcom.
AntiVirus Kit From 1st Aide Software.
Symantec Antivirus of the Mac version 3.0 or later.
Art of Computer Virus Research and Defense by Peter Szor (Feb. 2005).
"Process Execution Controls as a Mechanism to Ensure Consistency," by Eugene Bacic (1990).
"A Flexible Access Control Service for Java Mobile Code," by Corradi, et al., IEEE, pp. 356-365 (2000).
"Microsoft Authenticode Analyzed," by Rohit Khare (Jul. 22, 1996).
"Java Security: Whose Business Is It?," by Mark LaDue (1996).
"Mobile Code Security," by Rubin, et al., IEEE Internet, pp. 30-34, Dec. 1998.
"Protecting Data From Malicious Software," by Schmid, et al., Proceeding of the 18th Annual Computer Security Applications Conference, pp. 1-10, 2002.
"Security in the Large: Is Java's Sandbox Scalable?," by Zhong, et al. (Apr. 1998).
"A Domain and Type Enforcement UNIX Prototype," by Badger, et al., Proceedings of the Fifth USENIX UNIX Security Symposium, Salt Lake City, Utah (Jun. 1995), 15 pp.

(56) **References Cited**

OTHER PUBLICATIONS

"Heuristic Anti-Virus Technology," by Frans Veldman, Aug. 20, 2010.
"Standards for Security in Open Systems," by Warwick Ford (1989).
"Secure File Transfer Over TCP/IP," by Brown, et al., IEEE Region 10 Conference, (Nov. 11-13, 1992).
"Using CASE Tools to Improve the Security of Applications Systems," by Hosmer, et al. (1988).
"Miro: Visual Specification of Security," by Heydon, et al., IEEE Transactions on Software Engineering, vol. 16. No. 10, 13 pp. (Oct. 1990).
"An Evaluation of Object-Based Programming with Visual Basic," by Dukovic, et al. (1995).
"Visual Basic 5.0 Significantly Improved," by W. Dennis Swift, IEEE Antennas and Propagation Magazine, vol. 39, No. 3, 2 pp. (Jun. 1997).
"Development of an Object Oriented Framework for Design and Implementation of Database Powered Distributed Web Applications With the DEMETER Project as a Real-Life Example," by Goschka, et al. (1997).
"Detecting Unusual Program Behavior Using the Statistical Component of the Next-generation Intrusion Detection Expert System (NIDES)," by Anderson, et al. (May 1995).
"A Generic Virus Scanner in C++," by Kumar, et al., Proceedings of the 8th Computer Security Applications Conference, IEEE Press, (Sep. 17, 1992).
"A Model for Detecting the Existence of Software Corruption in Real Time," by Voas, et al., Computers & Security, 12, pp. 275-283 (1993).
"Protection Against Trojan Horses by Source Code Analysis," by Saito, et al., Electronics and Communications in Japan, Part 3, vol. 77, No. 1, 1994, Translated from Denshi Tsushin Gakkai Ronbunshi, vol. 76-A, No. 3, (Mar. 1993), pp. 466-472.
"Information Agents for Automated Browsing," by Dharap, et al. (1996).
"Static Analysis Virus Detection Tools for Unix Systems," by Kerchen, et al., 13th National Computer Security Conference, vol. 1, Washington, DC, Oct. 1-4, 1990, 17 pp.
"Managing Trust in an Information-Labeling System," by Blaze, et al. (Nov. 4, 1996).
"A Guide to the Selection of Anti-Virus Tools and Techniques," by Polk, et al. (Dec. 2, 1992).
"An Integrated Toolkit for Operating System Security," by Rabin, et al., Harvard University Aiken Computation Laboratory Technical Report, 55 pp. (Aug. 1988).
"Intel Launches Virus Counterattack," by Charles Bruno (Aug. 1992).
Intel LANProtect Software User's Guide, Intel Corporation (1992).
"Parents Can Get PC Cruise Control," by George Mannes (Jul. 1996).
"A New Technique for Detecting Polymorphic Computer Viruses," by Carey Nachenberg, Thesis, University of California, Los Angeles (1995).
"Heuristic Scanners: Artificial Intelligence," by Righard Zwienenberg, Virus Bulletin International Conference, 9 pp. (Sep. 20-22, 1995).
Intel LANProtect, 30-Day Test Drive Version User's Manual, 35 pp., 1992.
Earl Boebert's post to the greatcircle firewalls mailing list. Taken from http://www.greatcircle.com/lists/firewalls/archive/firewalls.199410 (Oct. 16, 1994).
CSL Bulletin: "Connecting to the Internet: Security Considerations." Taken from http://csrc.nist.gov/publications/nistbul/cs193-07.txt (Jul. 1993).
FAQ: Interscan Viruswall. Taken from http://web.archive.org/web/19970605050331/www.antivirus.com/faq/finterscanfaq.html (last updated Aug. 8, 1996).
"Network Security and SunScreen SPF-100: Technical White Paper," Sun Microsystems, 1995.
"Why Do We Need Heuristics?," by Frans Veldman, Virus Bulletin Conference, 6 pp. (Sep. 1995).
"Leading Content Security Vendors Announce Support for Check Point Firewall1 3.0; New Partners for Anti-Virus Protection, URL Screening and Java Security," Business Wire, Oct. 7, 1996, available at http://www.allbusiness.com/technology/computernetworks-computer-networksecurity 17274315-1.html#ixzz1gkbKf4g1.
"McAfee Introduces Web Shield; Industry's First Secure Anti-Virus Solution for Network Firewalls; Border Network Technologies and Secure Computing to Enter into Web Shield OEM Agreements," Business Wire, May 14, 1996, available at http://findarticles.comip/articles.mi_mOEIN1is_1996_May_14/ai_18283456/.
"Trend Micro Announces Virus and Security Protection for Microsoft Proxy Server; Also Blocks Java Applets, ActiveX," Business Wire, Oct. 29, 1996, available at http://www.thefreelibrary.com/Trend+Micro+announces+virus+and+security+protection+for+Microsoft-aO18810512.
Finjan's Opposition to Websense's Renewed Motion for Judgment as a Matter of Law, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
Declaration of Paul Batcher Re Websense, Inc.'s Proffer of Evidence Re Laches, dated Dec. 19, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
Opposition to Symantec's Motion for JMOL, dated Dec. 17, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA. No. 10-cv-593 (OMS) (Redacted Dec. 26, 2012).
"WWW Proxy to Cut Off Java," by Carl Claunch (Apr. 12, 1996), available at http://groups.google.com/group/muc.lists.firewalls/msg/38bb246b7eb3bec3.
"Combating Viruses Heuristically," by Franz Veldman, Virus Bulletin Conference 67-76 (Sep. 1993).
"Anti-Virus Tools and Techniques for Computer Systems," by Polk, et al., Noyes Data Corporation, Park Ridge, New Jersey, pp. 12-27, Copyright 1995 (with Cover and Copyright Pages—18 pp.).
"Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns," by LeCharlier, et al. (Jul. 2, 1995) (22 pp.).
"Blocking Java Applets at the Firewall," by Martin, et al., 11 pp. Available at http://www.cs.bu.edu/techreports/pdf/1996-026-Java-firewalls.pdf (1997).
"Virus: Detection and Elimination," by Rune Skardhamar, Academic Press, Inc., pp. 33-49, Copyright 1996 (with cover and copyright page—18 pp.).
Hruska, Jan, "Computer Viruses and Anti-Virus Warfare (Second Revised Edition)," Ellis Horwood, pp. 87-95, Copyright 1992 (with cover and copyright page—11 p.).
"Active Content Security," by Brady, et al. (Dec. 13, 1999).
"Low Level Security in Java," by Frank Yellin, 13 pp., [retrieved on Dec. 15, 2011], Retrieved from the Internet: mhtml:file//S:\DSFR_DATA/Data03/030052/01_Finjan_v._Symantec_(DDEL-00593)Prio . . . .
"Email With a Mind of its Own: The Safe-Tcl Language for Enabled Mail," by Nathaniel Borenstein.
"Mobile Agents: Are They a Good Idea?," by Chess, et al. (Dec. 21, 1994).
"Remote Evaluation," by Stamos, et al., ACM Transaction on Programming Languages and Systems, vol. 12, No. 4 (Oct. 1990).
"Active Message Processing: Messages as Messengers," by John Vittal (1981).
"Programming Languages for Distributed Computing Systems," by Bal, et al., ACM Computing Surveys, vol. 21, No. 3 (Sep. 1989).
"Scripts and Agents: The New Software High Ground," by John Ousterhout (Oct. 20, 1995).
The HotJava Browser: A White Paper, dated Dec. 15, 2011, 5 pp.
The Java Virtual Machine Specification, Sun Microsystems (Aug. 21, 1995).
"Security of Web Browser Scripting Languages: Vulnerabilities, Attacks and Remedies," by Anupam, et al. (Jan. 1998).
"ActiveX and Java: The Next Virus Carriers?," [retrieved on Jan. 2, 2011], 5 pp., Retrieved from the Internet: http://vx.netlux.org/lib/static/vdat/cpactive.htm.
"Gateway Level Corporate Security for the New World of Java and Downloadables" (1996).

(56) **References Cited**

OTHER PUBLICATIONS

"Practical Domain and Type Enforcement for UNIX," by Badger, et al. (1995), 12pp.
"A Sense of Self for Unix Processes," by Forrest, et al. (1996), 9 pp.
"Antivirus Scanner Analysis 1995," by Marko Helenius, 1995 [retrieved on Dec. 15, 2011], 7 pp., Retrieved from the Internet: http://www.cs.uta.fi/~cshcma/public/vru/documents/test1995.txt.
"State Transition Analysis: A Rule-Based Intrusion Detection Approach," by Ilgun, et al., IEEE Transactions on Software Engineering, vol. 21, No. 3 (Mar. 1995).
"Automated Detection of Vulnerabilities in Privileged Programs by Execution Monitoring," by Ko, et al. (1994).
"Execution Monitoring of Security-Critical Programs in Distributed Systems: A Specification-Based Approach," by Ko, et al. (1997).
"Classification and Detection of Computer Intrusions," by Sandeep Kumar, Thesis, Purdue University (Aug. 1995).
ThunderByte Anti-Virus Utilities User Manual (1995), 199pp.
Doyle, et al., "Microsoft Press Computer Dictionary," Microsoft Press, 2nd Edition, pp. 137-138, 1993.
"Process Execution Controls: Revisited," by Eugene Bacic (1990).
Schmitt, D.A., ".EXE Files, OS-2 Style," PC Tech Journal, vol. 6, No. 11, p. 76(13), Nov. 1988.
International Search Report for Application No. PCT/IB97/01626, dated May 14, 1998, 3 pp.
Supplementary European Search Report for Application No. EP 97 950351, dated Nov. 17, 2004, 2 pp.
File History for Canadian Application No. 2,275,771, 84 pp.
File History for European Application No. 97950351.3, 58 pp.
File History for Japanese Application No. 10-522345, 48 pp.
Order Construing the Terms of U.S. Pat. Nos. 6,092,194; 6,804,780; 7,058,822; 6,357,010; and 7,185,361, 4 pp., Dec. 11, 2007.
Plaintiff Finjan Software Ltd.'s Opening Claim Construction Brief, 38 pp., Sep. 7, 2007.
Defendant Secure Computing Corporation's Opening Claim Construction Brief, 46 pp., Sep. 7, 2007.
Plaintiff Finjan Software Ltd.'s Answering Claim Construction Brief (Public Version), 45 pp., Sep. 28, 2007.
Defendant Secure Computing Corporation's Responsive Claim Construction Brief (Public Version), 37 pp., Sep. 28, 2007.
Secure Computing Corporation's Disclosure of Prior Art Pursuant to 35 U.S.C. 282, 6 pp., Feb. 1, 2008.
Stang, David J., "Computer Viruses and Artificial Intelligence," Virus Bulletin Conference, pp. 235-257, Sep. 1995.
Johannsen, Magnus, "Java Security and a Firewall Extension for Authenticity Control of Java Applets," Thesis Proposal, Computer Science Department, University of Colorado at Colorado Springs, 5 pp., Jan. 29, 1997.
Joint Appendix of intrinsic and Extrinsic Evidence Regarding Claim Construction Briefing, vol. 1, Oct. 4, 2007.
Joint Appendix of intrinsic and Extrinsic Evidence Regarding Claim Construction Briefing, vol. 2, Oct. 4, 2007.
Final Joint Claim Construction Chart, Aug. 24, 2007.
Joint Post-Hearing Claim Construction Chart, Oct. 30, 2007.
Plaintiffs Trial Brief (Public Version), Jan. 14, 2008.
Marcionek, David, "A Complete ActiveX Web Control Tutorial," Available at http://www.codeproject.com/KB/COM/CompleteActiveX.aspx (2006).
Docket for *Finjan Software Ltd.* v. *Secure Computing Corporation, et al.*, U.S. District Court, District of Delaware (Wilmington), Civil Docket No. 1:06-cv-00369-GMS, 52 pp., retrieved on Oct. 24, 2008 from https://ecf.ded.uscourts.gov/cgibin/DktRpt.pl?6532182820_13655-L_567_0-1.
Docket for *Finjan Software Ltd.* v. *Aladdin Knowledge Systems, Inc., et al.*, U.S. District Court, District of Delaware (Wilmington), Civil Docket No. 1:08-cv-00300-GMS, 5 pp., retrieved on Oct. 24, 2008 from https://ecfded.uscourts.gov/cgi-bin-DktRpt.pl?994267838982431-L_567_0-1.
Firewall Toolkit (FWTK) 2.0 Beta Release, 1996.
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., Retrieved from the Internet: http://www.codeguru.com/Cpp/Cpp.cpp_mfc/parsing/article.php/c4093/.
Power, James, "Lexical Analysis," 4 pp., May 14, 2006, Retrieved from the Internet: http://www.cs.may.ie/~jpower/Courses/compilers/notes/lexical.pdf.
Sitaker, Kragen, "Rapid Genetic Evolution of Regular Expressions" [online], The Mail Archive, Apr. 24, 2004 (retrieved on Dec. 7, 2004), 5 pp., Retrieved on the Internet: http://www.mail-archive.com/kragen-tol@canonical.org/msg00097.html.
"Lexical Analysis: DFA Minimization & Wrap Up" [online], Fall, 2004 [retrieved on Mar. 2, 2005], 8 pp., Retrieved from the Internet: http://www.owlnet.rice.edu/~comp412/Lectures/L06LexWrapup4.pdf.
"Minimization of DFA" [online], [retrieved on Dec. 7, 2004], 7 pp., Retrieved from the Internet: http://www.cs.odu.edu/~toida/nerzic/390teched/regular/fa/min-fa.html.
"Algorithm: NFS → DFA" [online], copyright 1999-2001 [retrieved on Dec. 7, 2004], 4 pp., Retrieved from the Internet: http://rw4.cs.uni-sb.de/ganimal/GANIFA/page16_e.htm.
"CS 3813: Introduction to Formal Languages and Automata—State Minimization and Other Algorithms for Finite Automata," 3 pp., May 11, 2003, Retrieved from the Internet: http://www.cs.msstate.edu/~hansen/classes/3813fal101/slides/06Minimize.pdf.
Watson, Bruce W., "Constructing Minimal Acyclic Deterministic Finite Automata," [retrieved on Mar. 20, 2005], 38 pp., Retrieved from the Internet: http://www.win.tue.nl/~watson/2R870/downloads/madfa_algs.pdf.
Khare, R., "Microsoft Authenticode Analyzed" [online], Jul. 22, 1996 [retrieved on Jun. 25, 2003], 2 pp., Retrieved from the Internet: http://www.xent.com/FoRKarchive/smmer96/0338.html.
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, Retrieved from the Internet: activex.adsp.or.jp/inetsdk/help/overview.htm.
D. Grune, et al., "Parsing Techniques: A Practical Guide," John Wiley & Sons, Inc., New York, New York, USA, pp. 1-326, 2000.
Power, James, "Notes on Formal Language Theory and Parsing," National University of Ireland, pp. 1-40, 1999.
Scott, et al., "Abstracting Application-Level Web Security," ACM, pp. 396-407, 2002.
Non-Final Office Action, dated Jul. 23, 2012, for U.S. Appl. No. 13/290,708, 10 pp.
Amendment and Response to Office Action (dated Jul. 23, 2012) Under 37 C.F.R. 1.111, filed Oct. 23, 2012, for U.S. Appl. No. 13/290,708, 9 pp.
Final Office Action, dated Jan. 7, 2013, for U.S. Appl. No. 13/290,708, 7 pp.
Judgment, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (GMS).
Finjan's Opposition to Sophos' Renewed Motion for Judgment as a Matter of Law, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (GMS).
Finjan's Opposition to Symantec's Motion for Judgment as a Matter of Law at the Close of Evidence, dated Dec. 21, 2012, filed in *Finjan, Inc.* v. *Symantec Corp., Sophos, Inc., and Websense, Inc.*, CA No. 10-cv-593 (OMS).
Chang, Chia-Hsiang, "From Regular Expressions to DFA's Using Compressed NFA's," Oct. 1992, 243 pp., http://www.cs.nyu.edu/web/Research/Theses/chang_chi-hsiang.pdf.
Order Granting/Denying Request for Ex Parte Reexamination for U.S. Appl. No. 90/008,684, dated Aug. 28, 2007, 11 pp.
Request for Ex Parte Reexamination Under 35 U.S.C. 302-307 and 37 C.F.R. 1.510 for U.S. Appl. No. 90/008,684, dated Jun. 6, 2007, 49 pp.

(56) **References Cited**

OTHER PUBLICATIONS

Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,015, filed Oct. 7, 2013,38 pp.
Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,016, filed Oct. 7, 2013,48 pp.
Request for Ex Parte Reexamination for U.S. Appl. No. 90/013,017, filed Oct. 7, 2013, 69 pp.
Notice of Appeal for Civil Action No. 10-593-GMS, filed Sep. 27, 2013, 54 pp.
Order Denying Request for Ex Parte Reexamination, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,015, 16pp.
Order Granting Request for Ex Parte Reexamination, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,016, 17pp.
"Release Notes for the Microsoft ActiveX Development Kit," Nov. 1996 (10 pp.).
"Part 2: Java Vs. VBS—Breadth of Runtime Environments and Security Make Java a Good Choice for the Internet," Nikkei Electronics, pp. 212-221, No. 658, Mar. 25, 1996.
Lo, Raymond W., et al., "Towards a Testbed for Malicious Code Detection," from the Lawrence Livermore National Laboratory, 9 pp., Feb. 25- Mar. 1, 1991.
"Java Blocking," by C. Claunch, dated Sep. 25, 1996, available at http://groups.google.com/goup/muc.lists.firewalls/msg/2a5ec02c00a37071.
F. M. Avolio and M. J. Ranum, "A Toolkit and Methods for Internet Firewalls," In Technical Summer Conference, pp. 37-44, Boston, Massachusetts, Jun. 1994, USENIX, available at http://citeseer.ist.psu.edu/ranum94toolkit.html.
Petition for Writ of Certiorari to the Supreme Court of the United States in *Finjan, Inc.* v. *United States Patent and Trademark Office*, Apr. 2013.
Brief for the Respondent in Opposition to Petition for Writ of Cert in *Finjan, Inc.* v. *United States Patent and Trademark Office*, Jul. 2013.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Feb. 27, 2009, for U.S. Appl. No. 90/008,678, filed Jun. 6, 2007, 11 pp.
Amendment in Response to Non-Final Office Action dated May 8, 2008, filed Jul. 7, 2008, for U.S. Appl. No. 90/008,678, 27 pp.
Office Action in Ex Parte Reexamination, dated May 8, 2008, for U.S. Appl. No. 90/008,678, 18 pp.
Order Granting/Denying Request for Ex Parte Reexamination for U.S. Appl. No. 90/008,678, dated Aug. 28, 2007, 13 pp.
Request for Ex Parte Reexamination Under 35 U.S.C. 302-307 and 37 C.F.R. 1.510, for U.S. Appl. No. 90/008,678, dated Jun. 6, 2007, 67 pp.
Request for Ex Parte Reexamination Pursuant to 37 C.F.R. 1.501 for U.S. Appl. No. 90/009,175, dated Jun. 6, 2008, 75 pp.
Request for Inter Partes Reexamination Under 35 U.S.C. 311-318 for U.S. Appl. No. 95/001,836, 71 pp., dated Nov. 29, 2011.
Order Construing the Terms of U.S. Pat. Nos. 6,092,194 & 6,480,962, dated Feb. 29, 2012, 4 pp.
File History for U.S. Appl. No. 60/030,639, 33 pp., dated Nov. 8, 1996.
Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, dated Nov. 21, 2011, 246 pp.
Tanenbaum, Andrew S., "Computer Networks," Third Edition, Prentice Hall, Inc., ISBN 0-13-349945-6, Copyright 1996, 9 pp.
Flanagan, David, "Java in a Nutshell, A Desktop Quick Reference," 2nd Edition, O'Reilly & Associates, Inc., ISBN 1-56592-262-X, Copyright 1997, 4 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Proofpoint, Inc. and Armorize Technologies, Inc.*, Case No. 4:13-cv-05808, dated Dec. 16, 2013, 40 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Blue Coat Systems, Inc.*, Case No. 3:13-cv-03999, dated Aug. 28, 2013, 36 pp.
Complaint for Patent Infringement, *Finjan, Inc.* v. *Websense, Inc.*, Case No. 3:13-cv-04398, dated Sep. 23, 2013, 27 pp.
Docket for Case No. 3:13-cv-03999, *Finjan, Inc.* v. *Blue Coat Systems, Inc.*, Updated Dec. 5, 2013, 9 pp.
Docket for Case No. 3:13-cv-04398, *Finjan, Inc.* v. *Websense, Inc.*, Updated Nov. 25, 2013, 7 pp.
Zhong, et al., "Security in the Large: Is Java's Sandbox Scalable?," Seventh IEEE Symposium on Reliable Distributed Systems, pp. 1-6, Oct. 1998.
Hopwood, David, "A Comparison Between Java and ActiveX Security," Network Security, Elsevier Science Ltd., pp. 15-20, Dec. 1997.
Ungureanu, Victoria, et al., "A Policy-Based Access Control Mechanism for the Corporate Web," 16th Annual Computer Security Applications Conference (ACSAC '00), IEEE, Dec. 11, 2000.
Harn, Lein, et al., "A Software Authentication System for Information Integrity," Computers & Security, 1992.
Pei, Cao, Jin, et al., "Active Cache: Caching Dynamic Contents on the Web," Middleware '98, Springer London, Mar. 1998.
Bharadvaj, Harini, et al., "An Active Transcoding Proxy to Support Mobile Web Access," Dept. of Computer Engineering & Computer Science, University of Missouri-Columbia, Nov. 1998.
Bharadvaj, Harini, et al., "An Active Transcoding Proxy to Support Mobile Web Access," Proceedings, Seventeenth IEEE Symposium, IEEE, 1998.
Chi, Chi-Hung and Wu, Yin, "An XML-Based Data Integrity Service Model for Web Intermediaries," Proceedings, 7th IWCW, Aug. 14, 2002.
Article, "Java: The Inside Story, We Interview Java's Creators to Find What They Had in Mind," Sun World Online, Jul. 1995.
Giles, James, et al., "Authentication for Distributed Web Caches," Computer Security—ESORICS 2002, Springer Berlin, Heidelberg, Oct. 16, 2002.
Mandar, Raje, "Behavior-Based Confinement of Untrusted Applications," Dept. of Computer Science, UC Santa Barbara, TRCS 99-12, pp. 1-96, Jan. 1999.
Check Point Software Technologies Ltd., Archived webpages regarding Content Security and Partners, Check Point Software, 1997.
Hassler, Vesna and Then, Oliver, "Controlling Applets' Behavior in a Browser," Computer Security Applications Conference, Proceedings of the 14th Annual IEEE.
Seltzer, Margo I., et al., "Dealing With Disaster: Surviving Misbehaved Kernel Extensions," Proceedings of the USENIX 2nd Symposium on Operating Systems Design and Implementation, Seattle Washington, Oct. 1996.
Golvers, Luc, "Delivery and Installation of Software: Disputes and the Burden of Proof, Information Systems Security," Chapman & Hall, May 31, 1996.
Emin Gün Sirer, et al., "Design and Implementation of a Distributed Virtual Machine for Networked Computers," Proceedings of the Seventeenth ACM Symposium on Operating Systems Principles, pp. 202-216, Dec. 5, 1999.
Dr. Solomon's Anti-Virus Toolkit (prerelease), PC Magazine, Nov. 21, 1995.
"DRAFT, Internet Component Download," Microsoft Corporation, Jan. 1996.
Evans, David and Twyman, Andrew, "Flexible Policy-Directed Code Safety," Proceedings of the 1999 IEEE Symposium on Security and Privacy (IEEE), May 12, 1999.
Jackson, Keith, "FluShot+," Virus Bulletin, Nov. 1989.
"HotJava Browser Version 1.0 Frequently Asked Questions," Sun Microsystems, Inc., Nov. 5, 1997.
"HotJava Browser Version 1.0 Website Product Information," Sun Microsystems, Inc., Jan. 12, 1998.
"HotJava Browser: A White Paper," Sun Microsystems, Inc., 1994.
"IBM WebSphere Edge Server for Multiplatforms Administration Guide, Version 2.0," IBM, Dec. 2001.
Braswell, Byron, et al., "IBM WebSphere Edge Server: New Features and Functions in Version 2, IBM Redbooks," IBM, Apr. 2002.
Chockler, Gregory V., et al., "Implementing a Caching Service for Distributed CORBA Objects," Middleware 2000, Springer Berlin, Heidelberg, Apr. 4, 2000.
Claunch, Carl V., "Improved JavaScript and Java Screening Function," May 4, 1996.
"Integralis Announces MIMEsweeper for NT Version 2.1," Business Wire, Feb. 20, 1996.
"Integralis Launches the Internet's First Anti-Virus Protection System," M2 Press WIRE, Jun. 30, 1995.

(56) **References Cited**

OTHER PUBLICATIONS

"Integralis Mimesweeper Scans Internet Mail for Viri Nasties," Computer Business Review, Jul. 18, 1995.
Ethington, Brent, "Introducing Microsoft Windows 95," Microsoft Press, 1995.
Shin, Insik and Mitchell, John C., "Java Bytecode Modification and Applet Security," Stanford CS Tech Report, 1998.
McGraw, Gary and Felten, Edward W., "Java Security: Hostile Applets, Holes, and Antidotes," John Wiley & Sons, Inc., 1997.
"JavaSoft JAR File Format Specification Version 1.0," Sun Microsystems, Inc., Aug. 8, 1996z.
"Jingle Bells, Jingle Bells," Virus Bulletin, Jan. 1996.
Greenberg, Ross M., "Know Thy Viral Enemy," BYTE, Jun. 1989.
Vigil, Henry, "Making the Internet Safe for E-Commerce," Datamation, Oct. 1996.
Sibert, W. Olin, "Malicious Data and Computer Security," National Information Systems Security Conference, Oct. 22, 1996.
"MIMEsweeper Administrator Guide," Integralis, 1995.
"MIMEsweeper Review, PC User," PC User, Oct. 1996.
"MIMESweeper Review, Secure Computing," Secure Computing, Sep. 1996.
Small, Christopher, "MiSFIT: A Freely Available Tool for Building Safe Extensible Systems," Harvard Computer Science Technical Reports, Jul. 1996.
Tardo, Joseph and Valente, Luis, "Mobile Agent Security and Telescript," Compcon '96, 'Technologies for the Information Superhighway' Digest of Papers (IEEE), Feb. 25, 1996.
Chander, Ajay, et al., "Mobile Code Security by Java Bytecode Instrumentation," Computer Science Dept., Stanford University, and Dept. of Computer and Information Science, University of Pennsylvania.
Greenberg, Ross M., "MS-DOS Anti-Virus Tools and Techniques or How I Stopped Worrying and Learned to Trust in Debuggers," Virus Bulletin Conference, Sep. 1991.
Cheng, Kai and Kambayashi, Yahiko, "Multicache-Based Content Management for Web Caching," Proceedings of the First International Conference on Web Information Systems Engineering, Nov. 2000.
"Netscape Proxy Server, Chapter 9: How Caching Works," Netscape Communications Corporation, 2000.
Stallings, William, "Network and Internetwork Security Principles and Practice," Prentice Hall, 1995.
Weaver, Alfred C., et al., "Network Edge Services," 28th Annual Conference of the Industrial Electronics Society, IEEE.
Norton AntiVirus for Windows 95, PC Magazine, Nov. 21, 1995.
Anupan, Joshi, "On Proxy Agents, Mobility, and Web Access," Mobile Networks and Applications 5, pp. 233-241, 2000.
Riggs, Roger, et al., "Pickling State in the Java System," Proceedings of the USENIX 1996 Conference on Object-Oriented Technologies, Toronto, Ontario, Canada, Jun. 1996.
Jaeger, Trent and Rubin, Aviel D., "Preserving Integrity in Remote File Location and Retrieval," Proceedings of the Symposium on Network and Distributed System Security (IEEE), Feb. 22, 1996.
Press Release: "Trend Micro Ships InterScan AppletTrap—Enterprise Java and ActiveX Security Software," Business Wire, Mar. 23, 1999.
Thorn, Tommy, "Programming Languages for Mobile Code," ACM Computing Surveys, vol. 29, No. 3, pp. 213-239, 1997.
Necula, George C., "Proof-Carrying Code, Design and Implementation," School of Computer Science, Carnegie Mellon University, Sep. 1996.
Pandey, Raju and Hashii, Brant, "Providing Fine-Grained Access Control for Mobile Programs Through Binary Editing," TR-98-08, California University, Davis Dept. of Computer Science, 1997.
Iglio, Pietro, et al., "Rule-Based Filtering for Java Applets," Proceedings of the 14th Annual Security Applications Conference (IEEE), Dec. 1998.
Iglio, Pietro, et al., "Rule-Based Filtering for Java Applets," Computer Security Applications Conference, Proceedings of the 14th Annual IEEE, Dec. 1998.
Kirtland, Mary, "Safe Web Surfing With the Internet Component Download Service," Microsoft Systems Journal, Jul. 1996.
Erlingsson, Ulfar and Scheider, Fred B., "SASI Enforcement of Security Policies: A Retrospective," Proceedings of the 1999 Workshop on New Security Paradigms, ACM, 1999, Sep. 23, 1999.
Rubin, Aviel D., "Secure Distribution of Electronic Documents in a Hostile Environment," Computer Communications, Jun. 6, 1995.
Hsiao, Hui-I. and Brannon, Karen W., "Secure Information Caching on the Web," Advanced Web Technologies and Applications, Springer Berlin Heidelberg, 2004, Apr. 14, 2004.
Herbert, Andrew, "Secure Mobile Code Management Enabling Java for the Enterprise," Digitivity, Inc., May 1997.
Rosenblit, Moshe, "Secure Software Distribution," Network Operations and Management Symposium, 1994 (IEEE), Feb. 14, 1994.
Hashi, Brant, et al., "Securing Systems Against External Programs," Internet Computing IEEE, Nov. 1998.
Farmer, William M., et al., "Security for Mobile Agents: Authentication and State Appraisal," Computer Security—ESORICS 96, Springer Berlin Heidelberg, 1996, Sep. 25, 1996.
Wakeman, Ian, et al., "Signaling in a Component Based World," 1998 IEEE, 1998.
"Sun Microsystem's JavaSoft Java Beans 1.0 API Specification," Sun Microsystems, Inc., Oct. 15, 1996.
"SWEEP Website Product Overview—Client/Server Virus Detection: A Corporate Solution," Sophos, Dec. 11, 1997.
"SWEEP Website Product Overview—SWEEP Virus Detection for Windows NT," Sophos, Feb. 10, 1998.
"Symantec Announces First Native Java Virus Scanner and Leading Edge Virus Analysis Technology," Symantec Corp., Mar. 27, 1996.
Malkhi, Dahlia, et al., "The Design and Implementation of a Java Playground," AT&T Labs—Research, Oct. 10, 1997.
Sirer, Emin Gün, "The Kimerawork on Bytecode Verification," Department of Computer Science and Engineering, University of Washington, Internet: http://kimera.cs.washington.edu/overview.html, Oct. 30, 1997.
Evans, David E., "Thesis by David E. Evans, Policy-Directed Code Safety," Massachusetts Institute of Technology, Feb. 2000.
"ThunderBYTE Anti-Virus Utilities for Windows 95 (prerelease)," PC Magazine, Nov. 21, 1995.
Iliadis, J., et al., "Towards Secure Downloadable Executable Content: The Java Paradigm," Berlin Heidelberg, 1998.
Maheshwari, Anuj, et al., "TranSquid: Transcoding and Caching Proxy for Heterogenous Ecommerce Environments," Twelfth International Workshop on Research Issues in Data Engineering: Engineering E-Commerce/E-Business Systems, 2000.
"Trend Micro Product Paper, InterScan AppletTrap Eliminates Malicious Mobile Code," Trend Micro, Inc., Sep. 1999.
Swarup, Vipin, "Trust Appraisal and Secure Routing of Mobile Agents," DARPA Workshop on Foundations for Secure Mobile Code, Monterey, CA, USA, Feb. 17, 1997.
Feigenbaum, Joan and Lee, Peter, "Trust Management and Proof-Carrying Code in Secure Mobile-Code Applications," DARPA Workshop on Foundations for Secure Mobile Code, vol. 26, Mar. 27, 1997.
Chu, Yang-Hua, "Trust Management for the World Wide Web," Diss. Massachusetts Institute of Technology, Dept. of Electrical Engineering and Computer Science, Jun. 13, 1997.
Rabinovich, Michael and Spatscheck, Oliver, "WEB Caching and Replication," Addison Wesley, 2002.
Krishnamurthy, Balachander and Rexford, Jennifer, "Webware: Nightmare or Dream Come True?", National Information Systems Security Conference, Oct. 22, 1996.
"White Paper: HotJava Views 1.1 Software System Overview and Applet Integration," Sun Microsystems, Inc., Jun. 16, 2000.
Greer, Earl, "Wise Dr. Solomon's Fast Accurate in Catching Viruses," InfoWorld, Nov. 6, 1995.
Ghosh, Anup K., et al., "Wrapping Windows NT Software for Robustness," Digest of Papers, Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (IEEE), Jun. 18, 1999.
Docket for *Finjan, Inc.* v. *Fireeye, Inc.*, Case No. 4:13-cv-03133-SBA, dated Jul. 8, 2013.
Docket for *Finjan, Inc.* v. *Websense, Inc.*, Case No. 3:13-cv-04398-JSW, dated Sep. 23, 2013.

(56) **References Cited**

OTHER PUBLICATIONS

Twyman, Andrew R., "Flexible Code Safety for Win32," May 21, 1999.
Wagner, David A., "Janus: An Approach for Confinement of Untrusted Applications," 1999.
Symantec, "The Digital Immune System: Enterprise-Grade Anti-Virus Automation in the 21st Century," 2001.
Aladdin Knowledge Systems White Paper, "Complete Content Security for Internet Gateways and Mail Servers," eAladdin.com, 2002.
Ositis, Inc., "Ositis Announces eShield Platform," Business Wire Press Release, May 6, 2003.
Chen, "Poison Java," Aug. 1999.
Aladdin Knowledge Systems, Inc., "Privilege Secure Software Distribution: Comprehensive Electronic Software Licensing and Distribution Solution," Sep. 1999.
"EliaShim Announces Release of Unique Anti Virus Plug-In for Checkpoint's Firewall-1," PRNewswire Press Release, Feb. 10, 1997.
Wagner, David, et al., "A First Step Toward Automated Detection of Buffer Overrun Vulnerabilities," Feb. 3, 2000.
Bishop, Matt, "A Taxonomy of UNIX System and Network Vulnerabilities," May 1995.
Aladdin Knowledge System, "Aladdin eSafe Protect Desktop User's Manual," 1999.
Necula, George Ciprian, "Compiling With Proofs," Sep. 18, 1998.
Lambright, H. Dan., "Automated Verification of Mobile Code," Nov. 3, 1997.
Meyers, et al., "Automatic Detection of C++ Programming Errors: Initial Thoughts on a Lint ++," Aug. 1991.
Bishop, Matt and Dilger, Michael, "Checking for Race Conditions in File Access," Sep. 1995.
Aho, Alfred V., et al., "Compilers: Principles, Techniques, and Tools," Mar. 1988.
Kozen, Dexter, "Efficient Code Certification," Jan. 8, 1998.
Varvitsiotis, Angelos, et al., "EuroMed-Java: Trusted Third Party Services for Securing Medical Java Applets," Sep. 16, 1998.
Detlefs, David L., et al., "Extended Static Checking," Dec. 18, 1998.
Data Fellows Corp., "F-Secure Script Viruses Detector and Eliminator, Version 1.6," 1998.
Morrisett, Greg, et al., "From System F to Typed Assembly Language (Extended Version)," Nov. 21, 1997.
Reps, Thomas, et al., "Incremental Context-Dependent Analysis for Language-Based Editors," 1983.
Viega, John, et al., "ITS4: A Static Vulnerability Scanner for C and C ++ Code," Dec. 11, 2000.
Symantec, "Norton Antivirus for Firewalls 1.5 Eliminates Internet Viruses on IBM, CVP Firewalls Before They Enter Corporate Network," Oct. 20, 1999.
Johnson, S.C., "Lint, a C Program Checker," Jul. 26, 1978.
Ghosh, Anup K., "On Certifying Mobile Code for Secure Applications," Nov. 4, 1998.
Aladdin Knowledge System, "Safe Internet Connectivity for Home and Small Office," May 1999.
Bergeron, J., et al., "Static Analysis of Binary Code to Isolate Malicious Behaviors," Jun. 16, 1999.
Larochelle, David, et al., "Statically Detecting Likely Buffer Overflow Vulnerabilities," Aug. 13, 2001.
Radatti, Peter V., "The CyberSoft Virus Description Language," Feb. 1996.
Levy, Jacob Y., et al., "The Safe-Tcl Security Model," Jun. 1998.
Kipelainen, Pekka, "Tree Matching Problems With Applications to Structured Text Databases," Nov. 1992.
Trend Micro, "Trend InterScan WebManager 1.1," 1999.
McGraw, Gary, et al., "Understanding the Keys to Java Security—The Sandbox and Authentication," May 1, 1997.
Bishop, Matt, "Vulnerabilities Analysis," Sep. 1999.
Aladdin Knowledge System, "Web Archive eSafe.com (eSafe Protect Gateway)," Oct. 13, 1999.
Aladdin Knowledge Systems, "Web Archive of eSafe.com," Mar. 2000.
Bailey, "C ++ SoftBench CodeAdvisor 5.0," Dec. 2005.
Ribera, et al., "An Evaluation of Tools for Static Checking of C ++ Code," Feb. 2000.
ParaSoft, "Code Wizard," Nov. 1999.
Wilson, "A Look at PREfix by Intrinsa," Software Testing & Quality Engineering, May/Jun. 1999.
Internet Archive for www.prolint.com, "Revision History," Apr. 11, 2003.
Ghosh, et al., "An Automated Approach for Identifying Potential Vulnerabilities in Software," May 1998.
Internet Archive, "Secure Software Solutions: RATS," Jun. 6, 2001.
Christodorescu, Mihai and Jha, Somesh, "Static Analysis of Executables to Detect Malicious Patterns," Proceedings of the 12th USENIX Security Symposium, Aug. 4-8, 2003.
Roesch, Martin and Green, Chris, "Snort User Manual, Snort Release: 1.9.1," Feb. 26, 2003.
Kephart, J., et al., A Biologically Inspired Immune System for Computers, 1994.
Kephart, J., et al., Automatic Extraction of Computer Virus Signatures (1994).
Kephart, J., et al., Biologically Inspired Defenses Against Computer Viruses (1995).
S&S International, Dr. Solomon's Anti-Virus Toolkit for Windows and DOS (1995).
A. Solomon, Dr. Solomon's Virus Encyclopedia (1995).
Dr. Solomon's Anti-Virus Toolkit for Windows and DOS.
Dr. Solomon's Anti-Virus Toolkit for Netware.
Lunt, et al., "A Real-Time Intrusion Detection Expert System (IDES)," 1992.
"Sweep Windows NT User Manual," 1996.
"Sweep User Manual for Windows NT," 1995.
"Sweep User Manual for Windows 95," 1995.
Smaha, S., "Haystack: An Intrusion Detection System" (1988).
Department of Defense, "Department of Defense Trusted Computer System Evaluation Criteria" (1985).
"Check Point Software Delivers Breakthrough Security Advancements with Firewall-1 3.0; Industry's First to Offer Content Security Connection Control, Advanced Authentication and Encryption Integrated into an Open Enterprise Security Platform," Business Wire, 1996.
Venema, W., "TCP WRAPPER Network Monitoring, Access Control, and Booby Traps," In Proceedings of the Third Usenix UNIX Security Symposium (1992).
Radatti, "Computer Viruses in Unix Networks" (1995).
"Changing the Rules on Viruses," PC Magazine, Aug. 1994.
Chen, Steven, "PC Rx," PC Magazine, Mar. 16, 1993.
Sophos, "Intercheck," 1995.
Virus Bulletin, 20 pp., Nov. 1989.
Virus Bulletin, 24 pp., Aug. 1996.
Virus Bulletin, 32 pp., Jun. 1996.
Virus Bulletin, 24 pp., May 1995.
Virus Bulletin, 24 pp., Mar. 1995.
Virus Bulletin, 24 pp., Jun. 1995.
Virus Bulletin, 24 pp., Aug. 1995.
Virus Bulletin, 24 pp., Sep. 1995.
Virus Bulletin, 24 pp., May 1994.
Virus Bulletin, 24 pp., Mar. 1994.
Virus Bulletin, 24 pp., Feb. 1994.
Virus Bulletin, 24 pp., Apr. 1994.
Virus Bulletin, 24 pp., Jul. 1995.
Virus Bulletin, 24 pp., Apr. 1995.
Virus Bulletin, 24 pp., Dec. 1996.
Virus Bulletin, 24 pp., Oct. 1995.
Virus Bulletin, 24 pp., Jan. 1994.
Virus Bulletin, 24 pp., Jul. 1994.
Virus Bulletin, 24 pp., Aug. 1994.
Virus Bulletin, 24 pp., Feb. 1996.
Virus Bulletin, 24 pp., Feb. 1995.
Virus Bulletin, 24 pp., Nov. 1994.
Virus Bulletin, 24 pp., Nov. 1995.
Virus Bulletin, 24 pp., Nov. 1996.
Virus Bulletin, 24 pp., Dec. 1995.
Virus Bulletin, 24 pp., Dec. 1994.

(56) **References Cited**

OTHER PUBLICATIONS

Virus Bulletin, 24 pp., Oct. 1994.
Virus Bulletin, 24 pp., Apr. 1996.
Virus Bulletin, 24 pp., Sep. 1994.
Virus Bulletin, 24 pp., Jun. 1994.
Virus Bulletin, 24 pp., Sep. 1996.
Virus Bulletin, 24 pp., Jan. 1996.
Virus Bulletin, 24 pp., Mar. 1996.
Virus Bulletin, 24 pp., Jan. 1995.
Virus Bulletin, 24 pp., May 1996.
Virus Bulletin, 28 pp., Jul. 1996.
Virus Bulletin, 24 pp., Oct. 1996.
Mary R. Thompson, et al., "Certificate-Based Authorization Policy in a PKI Environment," Nov. 2003.
P. C. Cheng, "An Architecture for the Internet Key Exchange Protocol," 2001.
List of Secure Internet Programming Publications, [retrieved on Dec. 22, 2010], 4 pp., Retrieved from the Internet: www.cs.princeton.edu/sip/pub/index.php3.
"Synthesizing Fast Intrusion Prevention/Detection Systems From High-Level Specifications," by Sekar, et al. (1999).
"A Web Navigator With Applets in Caml," by Francois Ronaix, Fifth International World Wide Web Conference, Paris, France, May 6-10, 1996, 20 pp.
Slade, Robert, "Guide to Computer Viruses: How to Avoid Them, How to Get Rid of Them, and How to Get Help" (Apr. 1996).
Garfinkel, Simson and Spafford, Gene, "Practical Unix and Internet Security," O'Reilly & Associates, Inc., 1996.
ThunderBYTE Anti-Virus Utilities User Manual, ThunderBYTE B.V., Wijchen, The Netherlands, 1996.
"MCF: A Malicious Code Filter," by Lo, et al., 27 pp. (May 4, 1994).
"Towards a Testbed for Malicious Code Detection," by Lo, et al., Compcon Spring '91 Digest of Papers at 160-166, IEEE, 1991.
"Frequently Asked Questions—Java Security," Copyright 1995-1998 [retrieved on Dec. 15, 2011], 13 pp., Retrieved from the Internet: mhtml:file://S:\DFSR_DATA\Data03/030052\01_Finjan_v._Symantec(DDEL-00593)\Prio . . . .
Marcionek, David, "A Complete ActiveX Web Control Tutorial," Jun. 21, 2006 [retrieved on Sep. 29, 2014], 22 pp., Retrieved from the Internet: http://www.codeproject.com/Articles/14533/A-Complete-ActiveX-Web-Control-Tutorial.
Greenberg, Ross M., "Flu_Shot+, Version 1.81, A Form of Protection From Viral and Trojan Programs," Copyright 1988-1990, 54 pp.
"NetShield Version 1.61 Release Candidate for Novell NetWare v4.01 and Novell NetWare for OS/2 v4.01," McAfee Associates, Inc., Copyright 1994, 54 pp.
Balfanz, et al., "A Java Filter," 1997.
Ranum, et al., "A Toolkit and Methods for Internet Firewalls," 1994.
Pope, Nick, "Standards in Commercial Security," Designing Secure Systems, IEE Colloquium, Jun. 19, 1992.
Whyman, T., "X.400 Security Features," Security and Networks, IEE Colloquium, Jun. 25, 1990.
Donald E. Knuth, "The Art of Computer Programming," vol. 3 (1st Ed., 1973).
Donald E. Knuth, "The Art of Computer Programming," vol. 3, 2nd Ed., 1998).
"100 or So Books that Shaped a Century of Science," American Scientist, vol. 87, No. 6, pp. 542-544, 546, 549-550, 553 (Nov.-Dec. 1999).
Obituary of Frances E. Holberton, 84, "Early Computer Programmer," published Dec. 17, 2001.
http://getoutfoxed.com/compare ("Comparison"), Mar. 24, 2005.
http://getoutfoxed.com/options/search ("Different Engine Codes"), Apr. 26, 2005.
http://getoutfoxed.com/node/36 ("Abstract"), Mar. 19, 2005.
http://getoutfoxed.com/options/random_set ("Random Reports Configuration"), May 1, 2005.
http://getoutfoxed.com/node/32 ("Three Magic Ingredients"), Mar. 18, 2005.
Decision, dated Nov. 19, 2013, granting ex parte reexamination for U.S. Appl. No. 90/013,016, 17 pp.
Non-Final Office Action, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,016, 14 pp.
Decision, dated Dec. 6, 2013, granting ex parte reexamination for U.S. Appl. No. 90/013,017, 20 pp.
Non-Final Office Action, dated Dec. 6, 2013, for U.S. Appl. No. 90/013,017, 42 pp.
Final Office Action, dated Sep. 8, 2014, for U.S. Appl. No. 90/013,017, 51 pp.
Advisory Action, dated Oct. 23, 2014, for U.S. Appl. No. 90/013,017, 12 pp.
Examiner's Answer, dated Mar. 30, 2015, for U.S. Appl. No. 90/013,017, 117 pp.
Decision, dated Jan. 25, 2012, granting inter partes reexamination for U.S. Appl. No. 95/001,836, 77 pp.
Non-Final Office Action, dated Jan. 25, 2012, for U.S. Appl. No. 95/001,836, 17 pp.
Action Closing Prosecution dated Oct. 2, 2013, for U.S. Appl. No. 95/001,836, 61 pp.
Right of Appeal Notice, dated Jun. 27, 2014, for U.S. Appl. No. 95/001,836, 82 pp.
Examiner's Answer, dated Mar. 25, 2015, for U.S. Appl. No. 95/001,836, 4 pp.
Order Denying Request for Ex Parte Reexamination, dated Nov. 19, 2013, for U.S. Appl. No. 90/013,015, 16 pp.
Order Denying Request for Ex Parte Reexamination, dated Mar. 27, 2014, for U.S. Appl. No. 90/013,147, 26 pp.
Order Granting Request for Inter Partes Reexamination, dated Aug. 28, 2007, for U.S. Appl. No. 90/008,678, 13 pp.
Non-Final Office Action, dated May 8, 2008, for U.S. Appl. No. 90/008,678, 18 pp.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Feb. 27, 2009, 11 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,480,962, dated May 26, 2009, U.S. Appl. No. 90/008,678, 2 pp.
Order Granting Request for Ex Parte Reexamination, dated Aug. 28, 2007, for U.S. Appl. No. 90/008,684, 11 pp.
Non-Final Office Action, dated Dec. 19, 2008, for U.S. Appl. No. 90/008,684, 39 pp.
Final Office Action, dated Nov. 20, 2009, for U.S. Appl. No. 90/008,684, 53 pp.
Advisory Action, dated Feb. 3, 2010, for U.S. Appl. No. 90/008,684, 4 pp.
Examiner's Answer, dated Sep. 10, 2010, for U.S. Appl. No. 90/008,684, 86 pp.
Decision on Appeal, Examiner Affirmed, dated Jun. 6, 2011, for U.S. Appl. No. 90/008,684, 8 pp.
United States Patent and Trademark Office's Response to Appellant's Brief, dated Feb. 13, 2012, for U.S. Appl. No. 90/008,684.
Judgment of United States Court of Appeal for the Federal Circuit, dated Jan. 11, 2013, for U.S. Appl. No. 90/008,684, 2 pp.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated Jul. 26, 2013, for U.S. Appl. No. 90/008,684, 6 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,167,520, dated Sep. 6, 2013, U.S. Appl. No. 90/008,684, 2 pp.
Order Granting Request for Ex Parte Reexamination (U.S. Pat. No. 6,092,194), dated Aug. 28, 2008, for U.S. Appl. No. 90/009,175, 11 pp.
Notice of Intent to Issue Ex Parte Reexamination Certificate, dated May 28, 2009, for U.S. Appl. No. 90/009,175, 21 pp.
Ex Parte Reexamination Certificate for U.S. Pat. No. 6,092,194, dated Aug. 18, 2009, U.S. Appl. No. 90/009,175, 15 pp.
"Dynamic Detection and Classification of Computer Viruses Using General Behavior Patterns," by Morton Swimmer, in Proceedings of the 1995 Virus Bulletin Conference ("Swimmer 1995").
"Proof-Carrying Code," George C. Necula and Peter Lee. Technical Report CMU-CS-96-165, Nov. 1996.
"A Flexible Security Model for Using Internet Content," by Nayeem Islam et al., in Proceedings of the Sixteenth Symposium on Reliable Distributed Systems (1997).
"MIME Encapsulation of Aggregate Applet Objects" by Alireza Bahreman et al., IETF, Jun. 1996 ("Bahreman").

(56) **References Cited**

OTHER PUBLICATIONS

"An Immune System for Cyberspace," by Jeffrey O. Kephart et al., in Proc. of the IEEE SMC'97 ("Kephart 1997").
"A Generic Virus Detection Agent on the Internet," by Jieh-Sheng Lee et al., in the Proceedings of the Thirtieth Hawaii International Conference on System Sciences 1997 ("Lee").
"Understanding Heuristics: Symantec's Bloodhound Technology," Symantec White Paper Series vol. XXXIV (1997) ("Bloodhound WP").
"Understanding Symantec's Anti-virus Strategy for Internet Gateways," Symantec Enterprise Papers vol. XXX (1997) ("Gateways WP").
"Understanding and Managing Polymorphic Viruses," Symantec Enterprise Papers vol. XXX (1997) ("Striker WP").
"Symantec Announces Norton Antivirus for Firewalls for Maximum Protection From Internet-Borne Viruses," Symantec Press Release (1997) [retrieved on Feb. 23, 2015], located at http://www.symantec.com/about/news/release/article.jsp?prid=19970506_01 ("NAV for Firewalls Press Release").
Bruce Schneier, Applied Cryptography, John Wiley & Sons Inc. (1994) ("Applied Cryptography").
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance, Proceedings of the USENIX Security Symposium, San Francisco, CA (Aug. 5-9, 2002) ("Feamster USENIX").
Nick Feamster et al., Thwarting Web Censorship with Untrusted Messenger Discovery, 3rd Workshop on Privacy Enhancing Technologies, Dresden, Germany, Mar. 2003. ("Feamster Dresden").
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance (Presentation), 11th Usenix Security Symposium, Aug. 8, 2002 ("Feamster USENIX Presentation").
Nick Feamster et al., Infranet: Circumventing Web Censorship and Surveillance (Presentation), Harvard Kennedy School, Nov. 27, 2002 ("Feamster Kennedy Presentation").
ePolicy Orchestrator Version 2.0 Product Guide (2001) ("ePO 2.0 PG").
Common Updater Version 1.0 Configuration Guide (2002) ("CU 1.0 CG").
ePolicy Orchestrator Version 3.0 Product Guide (2003) ("ePO 3.0 PG").
ePolicy Orchestrator Version 3.0 Installation Guide (2003) ("ePO 3.0 IG").
William G.J. Halfond & Alessandro Orso, Combining Static Analysis & Runtime Monitoring to Counter SQL-Injection Attacks, Workshop on Dynamic Analysis, ACM, May 17, 2005 ("Halfond").
Yougang Song & Brett D. Fleisch, "Rico: a security proxy for mobile code," Elsevier, Jun. 2004 ("Song").
James Newsome and Dawn Song, Dynamic Taint Analysis for Automatic Detection, Analysis, and Signature Generation, May 2004.
D.P. Ghormley et al., SLIC: An Extensibility System for Commodity Operating Systems, USENIX Annual Technical Conference, Jun. 1998 ("Ghormley").
http://getoutfoxed.com/release/0.2 ("Download"), May 15, 2005.
http://getoutfoxed.com/node/43 ("Basic Architecture"), May 19, 2005.
http://getoutfoxed.com/about/search ("Search & Browsing"), Jun. 22, 2005.
http://getoutfoxed.com/blog/1 ("Stan's Blog"), Jun. 8, 2005.
http://getoutfoxed.com/node/37 ("Outfoxed System"), Jun. 22, 2005.
Insik Shin and John C. Mitchell, "Java Bytecode Modification and Applet Security," Feb. 21, 2001.
http://getoutfoxed.com/node/107 ("De-centralized Architecture"), Jun. 23, 2005.
http://getoutfoxed.com/node/42 ("Metadata Publication"), Jun. 23, 2005.
http://getoutfoxed.com/help/options ("Options"), Jul. 27, 2005.
http://getoutfoxed.com/node/114 ("Security"), Oct. 24, 2005.
http://getoutfoxed.com/screenshots ("Screenshots Page"), May 15, 2005.
http://getoutfoxed.com/files/screenshots/spyware_search.htm ("Spyware Page"), May 14, 2005.
http://getoutfoxed.com/files/screenshots/osnabrueck_search.htm ("Osnabrueck Screenshot"), May 15, 2005.
http://getoutfoxed.com/files/screenshots/boingboinggoogle.htm ("Preshrunk Screenshot"), Jun. 22, 2005.
http://getoutfoxed.com/files/screenshots/warcraft_search.htm ("Warcraft Screenshot"), Jun. 22, 2005.
http://getoutfoxed.com/files/screenshots/claria_dangerous.png ("Claria Screenshot") [retrieved Feb. 23, 2015].
http://getoutfoxed.com/files/screenshots/a9_random.jpg ("A9 Screenshot") [retrieved Feb. 23, 2015].
http://getoutfoxed.com/files/screenshots/google_random.png ("Google Random Screenshot") [retrieved Feb. 23, 2015].
http://getoutfoxed.com/files/screenshots/boingboinggoogle2.htm ("Google BoingBoing Screenshot"), Dec. 16, 2005.
http://getoutfoxed.com/files/screenshots/google_searchl.png ("Google Weatherbug Screenshot") [retrieved Feb. 23, 2015].
http://getoutfoxed.com/files/screenshots/boingboinggoogle.png ("Google BoingBoing Screenshot 2") [retrieved Feb. 23, 2015].
http://getoutfoxed.com/files/screenshots/process_list.png ("Process List Screenshot"), Dec. 16, 2005.
http://getoutfoxed.com/files/screenshots/randomset_google.png ("Google Random Screenshot 2"), Dec. 16, 2005.
BetterSearch Homepage (Aug. 29, 2005) ("BetterSearch Homepage").
BetterSearch Screenshots Page (Jul. 20, 2005) ("BetterSearch Screenshots").
BetterSearch Installation Page (Jul. 28, 2005) ("BetterSearch Installation").
BetterSearch News Page (Aug. 26, 2005) ("BetterSearch News").
BetterSearch Changelog Page (Aug. 26, 2005) ("BetterSearch Changelog").
BetterSearch FAQ Page (Aug. 27, 2005) ("BetterSearch FAQ").
George C. Necula, "Proof-carrying code." In Proceedings of the 24th ACM Symposium on Principles of Programming Languages, Paris, France, Jan. 1997.
Encapsulating Mobile Objects, by Hermann Hartig et al., in the Proceedings of the 17th International Conference on Distributed Computing Systems, May 1997 ("Hartig").
Christopher Small, "A Tool for Constructing Safe Extensible C++ Systems," In Proceedings of the Third USENIX Conference on Object-Oriented Technologies (1997).
"Symantec's Norton AntiVirus 4.0 Delivers Multi-Platform Support," Symantec (1997) (Sep. 23, 1997) http://symantec.com/about/news/release/article.jsp?prid=19970923_01.
"Symantec's AntiVirus Research Center Responds to Mutating Macro Viruses With Bloodhound-Macro Technology," Symantec (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970923_03.
"Symantec Announces Norton Internet Email Gateway at Internet World," Symantec Press Release (1996) http://www.symantec.com/about/news/release/article.jsp?prid=19961211_03.
"Symantec Ships Norton AntiVirus for Internet Email Gateways," Symantec Press Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970224_01.
"Symantec Announces Norton Antivirus for Firewalls for Maximum Protection From Internet-Borne Viruses," Symantec Press Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970506_01 ("NAV for Firewalls Press Release").
"Symantec, The Worldwide Leader in Antivirus, Ships Norton Antivirus for Firewalls," Symantec Press Release (1997) http://www.symantec.com/about/news/release/article.jsp?prid=19970714_01.
Check Point, "Check Point Software Delivers Breakthrough Security Advancements with FireWall-1.3.0," Oct. 7, 1996, http://www.checkpoint.com/press/1996/30release.html.
Virus Bulletin, International Conference, Sep. 20-22, 1995.
Application as Filed for U.S. Appl. No. 60/711,012, filed Aug. 24, 2005.
Application as Filed for U.S. Appl. No. 60/711,972, filed Aug. 25, 2005.
Application as Filed for U.S. Appl. No. 60/735,772, filed Nov. 10, 2005.

(56) **References Cited**

OTHER PUBLICATIONS

Application as Filed for U.S. Appl. No. 60/581,857, filed Jun. 21, 2004.
Gene H. Kim and Eugene H. Spafford, "The Design and Implementation of Tripwire: A File System Integrity Checker," 23 pp., 1993.
Mark Kennedy, "Script-Based Mobile Threats," Virus Bulletin Conference, Sep. 2000.
Dr. Vesselin Bontchev and Katrin Tocheva, "Macro and Script Virus Polymorphism," Proceedings of the Twelfth International Virus Bulletin Conference, pp. 406-437, 2002.
Chris Shiflett, "PHP Security," O'Reilly Open Source Convention, Jul. 26, 2004.
"Mod-Security Reference Manual, v. 1.8.5," Oct. 26, 2004.
Belden Menkus, "A Secure Electronic Document Audit Trail Product," Jun. 1995.
David R. Safford, Douglas Lee Schales, and David K. Hess, "The TAMU Security Package: An Ongoing Response to Internet Intruders in an Academic Environment," Proceedings of the Fourth USENIX UNIX Security Symposium, Santa Clara, California, Oct. 4-6, 1993.
F-Prot, "Our Technology—F-Prot Antivirus" [online], Copyright 2014 [retrieved on Jun. 7, 2015], Retrieved from the Internet: http://www.f-prot.com/technology.
"Parasoft Introduces CodeWizard" [online], May 5, 1996 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/430/.
"CodeWizard for Windows 95 and Windows NT Available Soon!" [online], Oct. 1, 1996 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/429/.
"Parasoft Announces CodeWizard for Windows" [online], Mar. 3, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/428/.
"CodeWizard Supports Microsoft Visual C++ v. 5.0" [online], Mar. 24, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/426/.
"Parasoft Releases Preliminary Version of CodeWizard for Java" [online], Mar. 24, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/427/.
"Parasoft Releases CodeWizard v. 1.1 Beta" [online], May 23, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/425/.
"Parasoft Announces Pre-Release Version of CodeWizard for Java for Windows NT/95" [online], Jun. 2, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/424/.
"Parasoft Releases CodeWizard v. 1.1" [online], Jul. 3, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/423/.
"Parasoft Announces CodeWizard v. 1.5, Preventative Medicine for Error-Prone Code" [online], Jul. 28, 1997 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/422/.
"CodeWizard—Parasoft Forum" [online], Apr. 27, 2006 (downloaded May 8, 2015), Retrieved From: http://forums.parasoft.com/index.php?showtopic=996&mode=threaded&pid=2560.
"Parasoft Introduces CodeWizard Plus The Intelligent Development Solution for C++ and Java Developers" [online], Sep. 2, 1997 (downloaded May 7, 2015), Retrieved From: http://www.parasoft.com/press/421/.
"Parasoft Launches Newest Version of Award-Winning Automatic Error Detection Tool for C/C++, Insure++, v. 4.1" [online], Jun. 8, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/315/.
"Parasoft Awarded Logo Trademark for C/C++ Test Coverage Tool, TCA" [online], Aug. 31, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/313/.
"Parasoft Tools Insure++ and CodeWizard Offer Needed Stability, Reliability for Microsoft Visual C++ 6.0 Developers" [online], Oct. 28, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/311/.
"Parasoft's Insure++ and CodeWizard Now Support 64-Bit Development on HP11 and SGI64" [online], Dec. 7, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/308/.
"Newly Enhanced Versions of CodeWizard for C++ and Java Coming Soon" [online], Nov. 4, 1997 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/420/.
"Parasoft Releases Beta Version of CodeWizard for HTML, a Web Error Prevention Tool for Use With WebKing" [online], Feb. 1, 1999 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/416/.
"Internet Error Prevention and Detection: How Dysfunctional Are the Fortune 100 Web Sites?" [online], Nov. 8, 1999 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/739/.
"Parasoft Director of Technology, Dr. Michael Aivazis, to Speak at LinuxWorld Conference and Expo" [online], Jan. 24, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/655/.
"Parasoft Presents Solutions for Object-Oriented Embedded Development" [online], Feb. 21, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/654/.
"Parasoft to Showcase Solutions for Web Development At Internet World" [online], Mar. 27, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/735/.
"New Versions of CodeWizard for C++ and Java Available Now!" [online], Dec. 4, 1997 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/419/.
Adam Kolawa, Ph.D., "Parasoft & Innovation" [online], Apr. 2008 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/company/20-year-retrospectove/.
"Parasoft Extends Support to Windows 2000" [online], Apr. 10, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/645/.
"Boeing Chooses Parasoft's CodeWizard to Automate Coding Standard" [online], Apr. 17, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/414/.
"Parasoft and DCS IP, LLC Announce Sales Partnership" [online], May 2, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/572.
"Parasoft Releases CodeWizard v 3.1 for C and C++ " [online], May 3, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/413/.
"Parasoft to Present Newest Upgrade of WebKing at WebLA" [online], May 15, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/733/.
"Parasoft Releases CodeWizard Version 3.2" [online], Sep. 25, 2000 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/412/.
"Parasoft Releases New Version of Java Tool CodeWizard for Java v. 1.1" [online], May 26, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/418/.
"IMA Reduces the Time and Cost of Regulatory Compliance Through Static Analysis," Parasoft, 2 pp.
"Parasoft Forum" [online], Apr. 8, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showtopic=206&mode=threaded&pid=415.
"Parasoft C/C++ Test" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/product/epptest/.
"Parasoft, This Forum" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showforum=16.
"Parasoft Forum, Website" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showforum=17.
"Parasoft Forum, Custom Link Check" [online], Jun. 22, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t275.html.
"Parasoft Forum, How Can I Check My Codes Without Compiler" [online], Jun. 9, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t264.html.

(56) **References Cited**

OTHER PUBLICATIONS

"Parasoft Forum, CodeWizard Installation Problem" [online], Aug. 1, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiverison.pho/t306.html.
"Parasoft Launches Newest Version of Award-Winning Automatic Software Detection Tool for C/C++, Insure++ v. 4.1 for Windows" [online], Jun. 22, 1998 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/314/.
"Parasoft's CodeWizard for Java Avoid "Nasty" Surprises in Their Programs" [online], Sep. 21, 1998 [retrieved on May 7, 2015], Retrieved From the Internet: http://www.parasoft.com/press/417/.
"Parasoft Forum, CodeWizard" [online], Apr. 27, 2006 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiversion/index.php/t996.html.
"Parasoft Forum, WebKing and Usability Test" [online], Feb. 23, 2005 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/lofiverison/index.php/t500.html.
"Parasoft Forum, Repeated Element Regression Control" [online], Jun. 9, 2004 [retrieved on May 8, 2015], Retrieved From the Internet: http://forums.parasoft.com/index.php?showtopic=262.
"Parasoft Sitemap" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/sitemap.
"Parasoft to Exhibit Solutions for Object-Oriented Embedded Development at Embedded Systems Conference, Boston" [online], Jun. 26, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/638/.
"Parasoft to Showcase Error Prevention and Error Detection Software at Upcoming Embedded Systems Conference" [online], Sep. 18, 2000 [retrieved on May 8, 2015], Retrieved From the Internet: http://www.parasoft.com/press/628/.
"Parasoft Launches New Automated Database Testing Tool to Validate Application-Database Interaction" [online], Nov. 19, 2002 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1010/.
"Parasoft Joins Eclipse" [online], Dec. 10, 2002 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1023/.
"Parasoft WebKing Earns IBM Ready for WebSphere Validation" [online], Jan. 29, 2003 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1033/.
"Top Companies Choose Parasoft to Improve Software Quality and Reliability" [online], Jul. 2, 2003 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1189/.
"Parasoft Extends Support to Software Developers in the Automotive Industry" [online], Jan. 7, 2004 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/press/1336/.
"Parasoft Forum, ANT Task for CodeWizard" [online], Aug. 12, 2004 [retrieved on May 8, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=319.
"Parasoft's CodeWizard v. 4.0 Enhances Software Error Prevention Capabilities" [online], Aug. 27, 2001 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/press/411/.
Arthur Hicken, "Service Virtualization, Development Resting, API Testing Tips, False Positives in Statuc Code Analysis" [online], Feb. 21, 2013 [retrieved on May 8, 2015], Retrieved from the Internet: http://blog.parasoft.com/bid/113524/False-Positive-in-Static-Code-Analysis.
"Parasoft Forum, CodeWizard Forum, Where to Post CodeWizard Topics?" [online], Apr. 6, 2004 [retrieved on May 7, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=201.
"CodeWizard User Testimonial (Suzette LaGray)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/346/.
"CodeWizard User Testimonial (Mike LeBlanc, SKY Computers, Inc.)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/348/.
"CodeWizard User Testimonial (Javier Borrajo, Telefonica Investigacion & Desarolio)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/351/.
"CodeWizard User Testimonial (Gura Rao, Data Research Associates)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/353/.
"CodeWizard User Testimonial (Martin A. Bailey, Richardson Communications Group)" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/testimonial/355/.
"Parasoft Forum, Download CodeWizard" [online], Jul. 25, 2005 [retrieved on May 8, 2015], Retrieved from the Internet: http://forums.parasoft.com/index.php?showtopic=682.
"Parasoft, Awards & Recognitions" [online], Copyright 2015 [retrieved on May 8, 2015], Retrieved from the Internet: http://www.parasoft.com/company/awards/.
Parasoft, "Software Testing Tools—From Static Analysis to Service Virtualization" [online], Copyright 2015 [retrieved on May 7, 2015], Retrieved from the Internet: http://www.parasoft.com/products/.
"Parasoft CodeWizard" [online], Copyright 1996-2001 [retrieved on Jun. 5, 2015], Retrieved from the Internet: http://char.tuiasi.ro/doace/www.parasoft.com/products/wizard.
Ben Crowder, "CodeWizard for Linux" [online], Nov. 1, 1999 [retrieved on Jun. 5, 2015], Linux Journal, Retrieved from the Internet: http://www.linuxjournal.com/article/3615.
Tom Bragg, "Parasoft's CodeWizard for Java" [online], Dec. 18, 1999 [retrieved on Jun. 5, 2015], Retrieved from the Internet: http://adtmag.com/articles/1999/12/18/parasofts-codewizard-for-java-aspx.
U.S. Pat. No. 6,154,844 Prosecution History Excerpts.
MARC Record ("A Flexible Security Model for Using Internet Content"), 1997.
Joint Claim Construction and Pre-Hearing Statement Pursuant to Local Rule 4-3, dated Mar. 16, 2015, submitted by Finjan, Inc. and Symantec Corporation, Case No. 3-14-CV-02998-HSG.
Plaintiff Finjan, Inc.'s Opening Claim Construction Brief, dated Apr. 20, 2015, 30 pp., Case No. 14-CV-02998-HSG.
Wahbe, et al., "Efficient Software-Based Fault Isolation," ACM SIGOPS (Special Interest Group on Operating Systems) Symposium on Operating Systems Principles, Dec. 1993.
Jack W. Davidson, et al., "Safe Virtual Execution Using Software Dynamic Translation," 18th Annual Computer Security Applications Conference, Las Vegas, Nevada, Dec. 2002
Random House Webster's College Dictionary, Second Edition, Random House, New York, Copyright 1999, Cover, Copyright Page and p. 339 (Database).
Webster's New World Dictionary of Computer Terms, Fourth Edition, Prentice Hall, New York, NY, Copyright 1992, and p. 95 (Database).
U.S. Appl. No. 60/205,591, filed May 17, 2000, 9 pp.
Reexamination Excerpts for U.S. Appl. No. 90/013,017, 9 pp.
File History Excerpts from U.S. Appl. No. 13/290,708 (Now Patent No. 8,677,494), 29 pp.
U.S. Appl. No. 12/174,592 (Petition to Accept Unintentionally Delayed Claim of Priority filed on Dec. 20, 2011 (denied) and Jan. 25, 2012 (granted).
U.S. Appl. No. 12/174,592 (applicants' response and amendment filed on Jul. 19, 2011).
MARC Record ("A Sense of Self for Unix Processes"), 1996.
Virus Bulletin Conference, Proceedings of the Fifth International Virus Bulletin Conference, Sep. 1995 (Parts 1-3).
MARC Record ("Dynamic Detection and Classification of Computer Viruses Using General Behaviour Patterns"), 1995.
Duncan, Ray, "Advanced MS DOS," Microsoft Press, Redmond Washington, Copyright 1986, cover, Copyright Page, and pp. 272-282.
"Digital Signature Initiative Proposal," Aug. 1, 1996.
James W. Moore, "The Use of Encryption to Ensure the Integrity of Reusable Software Components," IEEE, 1994.
Judson D. Weeks, et al., "CCI-Based Web Security: A Design Using PGP," W3C, Dec. 13, 1995.

(56) **References Cited**

OTHER PUBLICATIONS

Burkhard Wiegel, "Secure External References in Multimedia Email Messages," Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, Mar. 14-15, 1996.

Ofer Maor, et al., "SQL Injection Signatures Evasion," Imperva Application Defense Center, Apr. 2004.

Ryoichi Sasaki, et al., "Security and Privacy in the Age of Ubiquitous Computing," IFIP TC11 20th International Information Security Conference, Chiba, Japan, May 30-Jun. 1, 2005.

Francisco Fernandez, "Heuristic Engines," Virus Bulletin Conference, Sep. 2001.

Alex Shipp, "Heuristic Detection of Viruses Within Email," Virus Bulletin Conference, Sep. 2001.

Klaus Julisch, et al., "Detection of Intrusions and Malware, and Vulnerability Assessment," Second International Conference, DIMVA 2005, Vienna, Austria, Jul. 7-8, 2005.

Christopher Kruegel, et al., "On the Detection of Anomalous System Call Arguments," Computer Security—ESORICS 2003, 8th European Symposium on Research in Computer Security, Gjovik, Norway, Oct. 13-15, 2003.

John Graham-Cumming, "The Waxing and Waning of Spammers' Trickery," Proceedings of the 14th Virus Bulletin International Conference, Chicago, Sep. 29-Oct. 1, 2004.

Eric Lippert's Blog, "Eval is Evil, Part Two," Nov. 4, 2003.

* cited by examiner

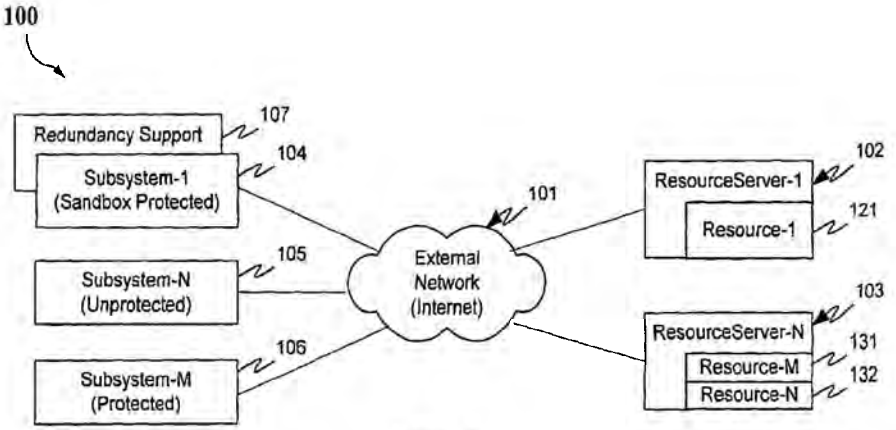


FIG. 1a

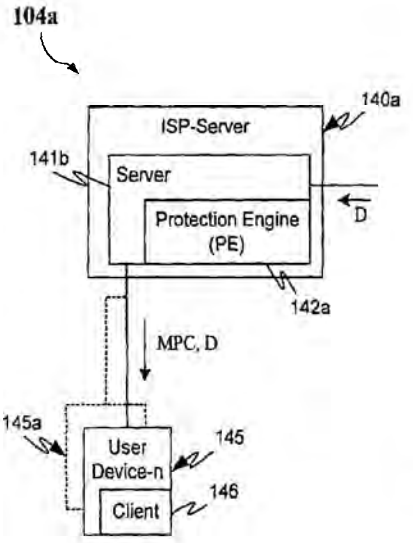


FIG. 1b

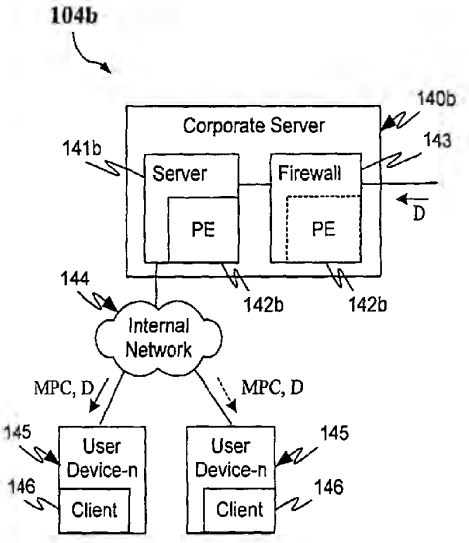


FIG. 1c

Case 3:17-cv-00183-BAS-NLS Document 1-6 Filed 07/01/16 PageID.188 Page 18 of 38

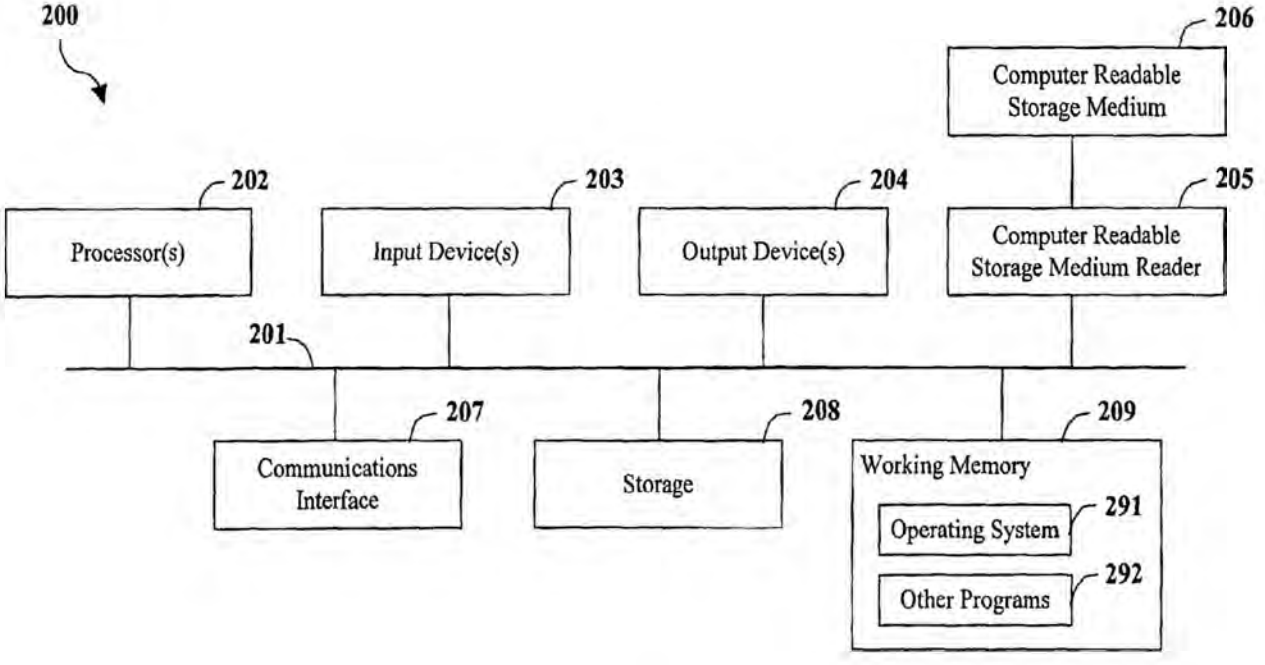


FIG. 2

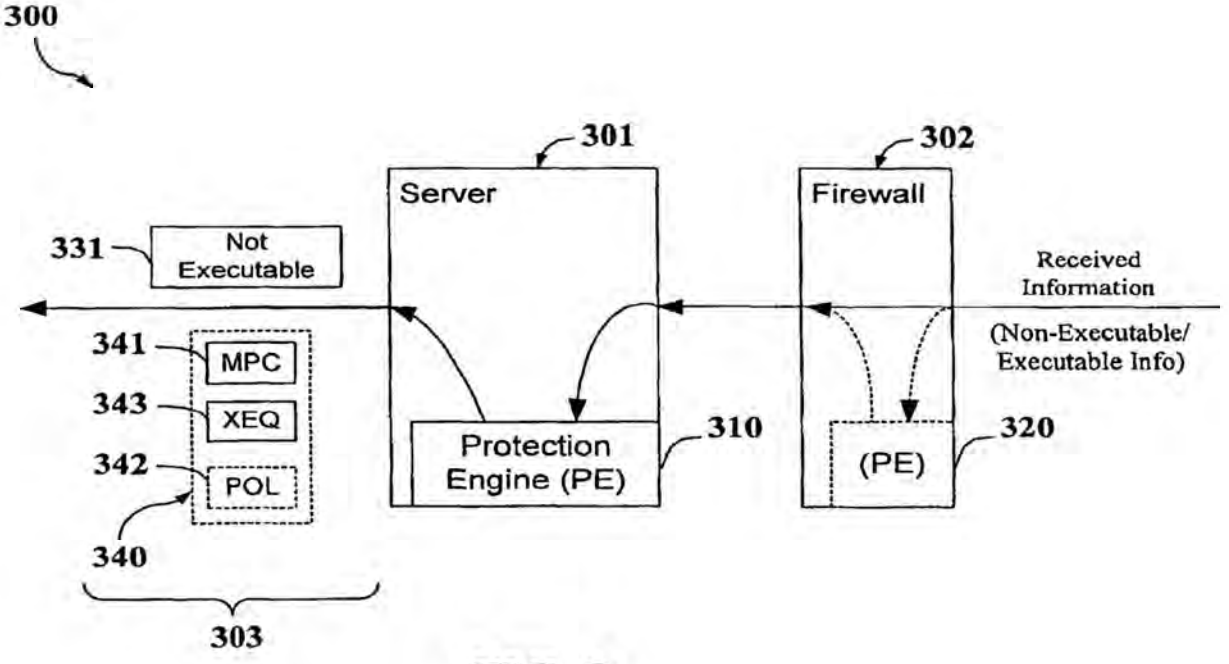

300
301
Server
302
Firewall
331
Not Executable
Received Information
(Non-Executable/ Executable Info)
341
MPC
343
XEQ
342
POL
340
303
Protection Engine (PE)
310
(PE)
320

FIG. 3

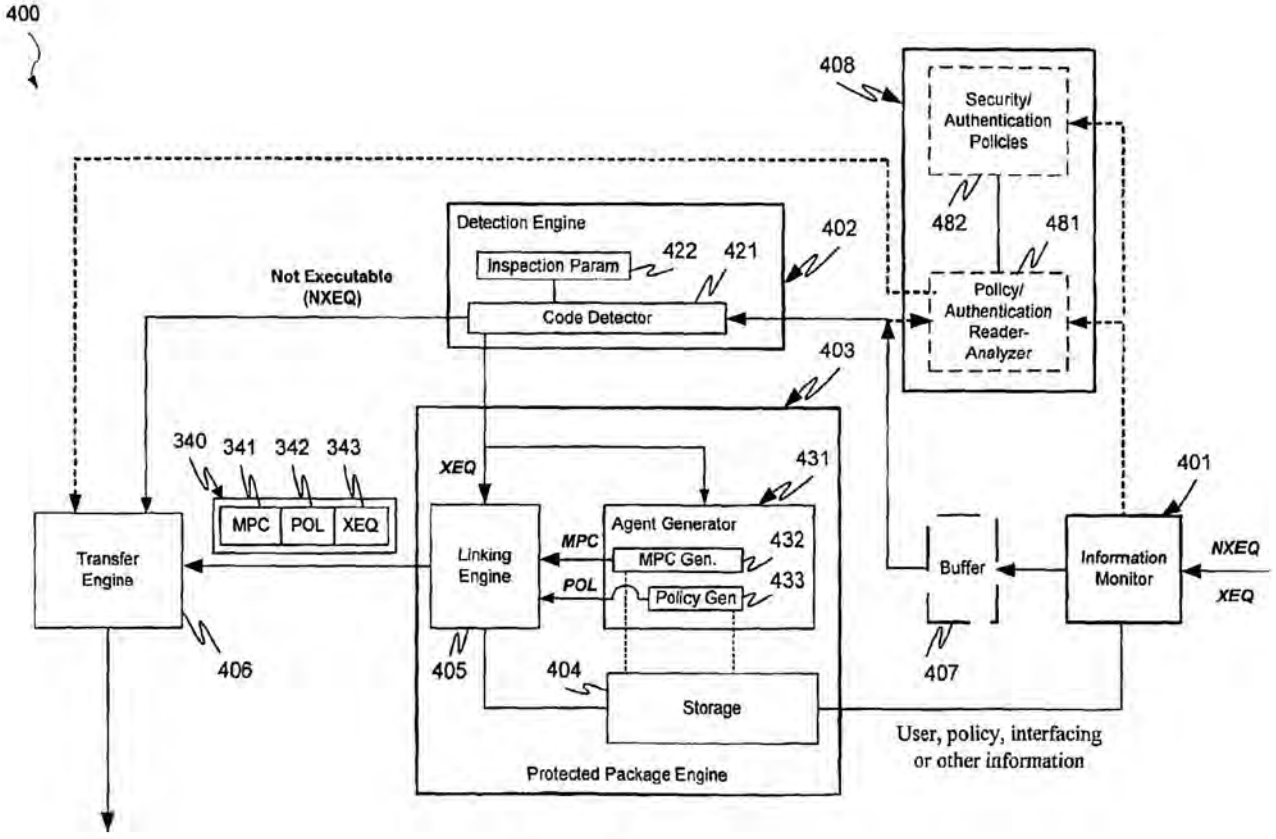

400
408
Security/ Authentication Policies
482
481
Policy/ Authentication Reader-Analyzer
Detection Engine
402
Inspection Param
422
421
Code Detector
Not Executable (NXEQ)
403
340
341
342
343
MPC
POL
XEQ
XEQ
431
Agent Generator
MPC
MPC Gen.
432
POL
Policy Gen
433
Transfer Engine
Linking Engine
406
405
404
Storage
Protected Package Engine
Buffer
407
401
Information Monitor
NXEQ
XEQ
User, policy, interfacing or other information

FIG. 4

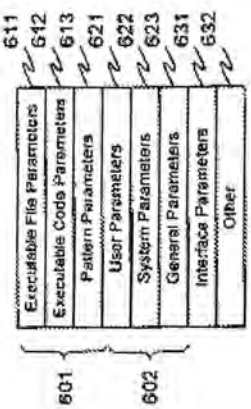


FIG. 6a

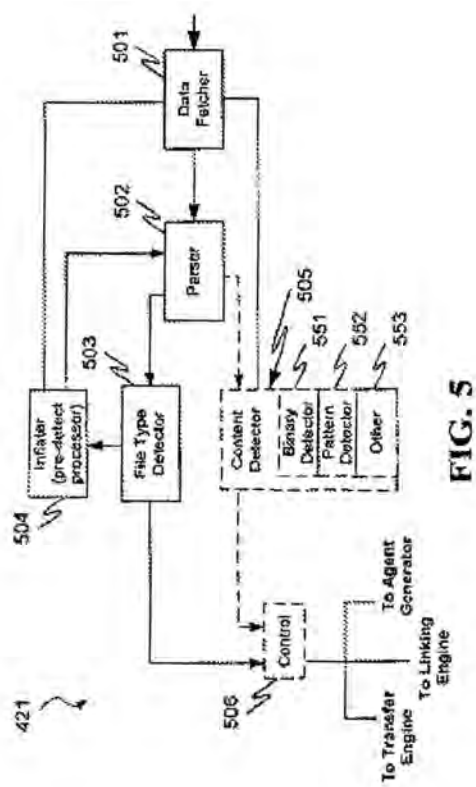


FIG. 5

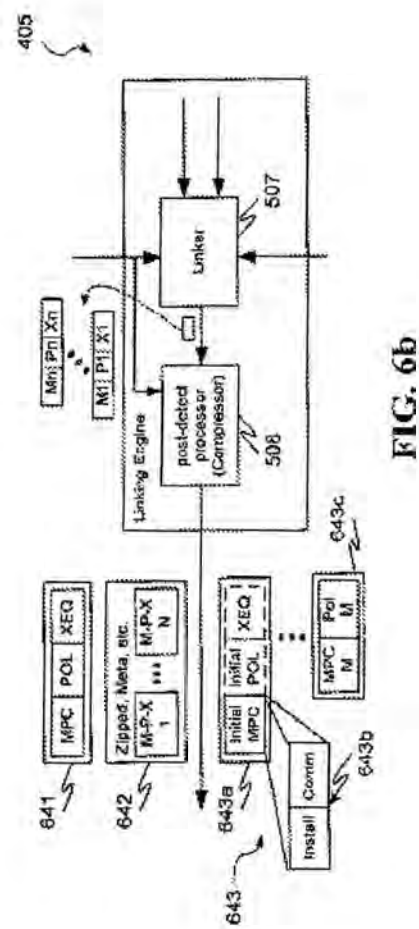


FIG. 6b

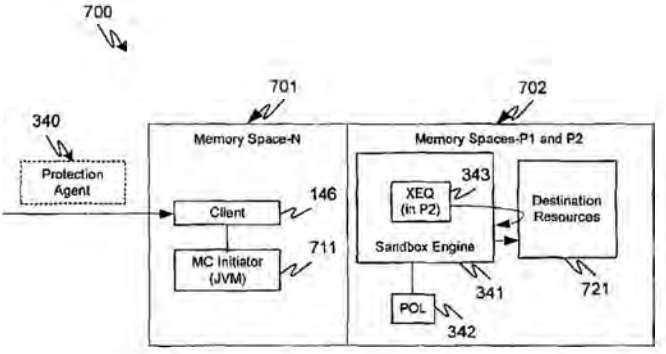


FIG. 7a

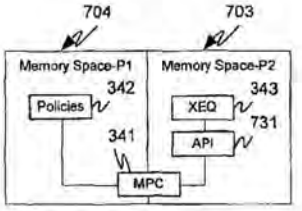


FIG. 7b

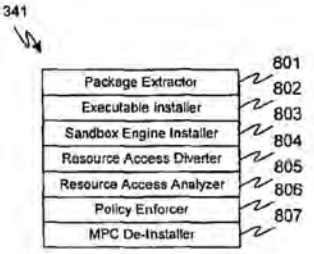


FIG. 8

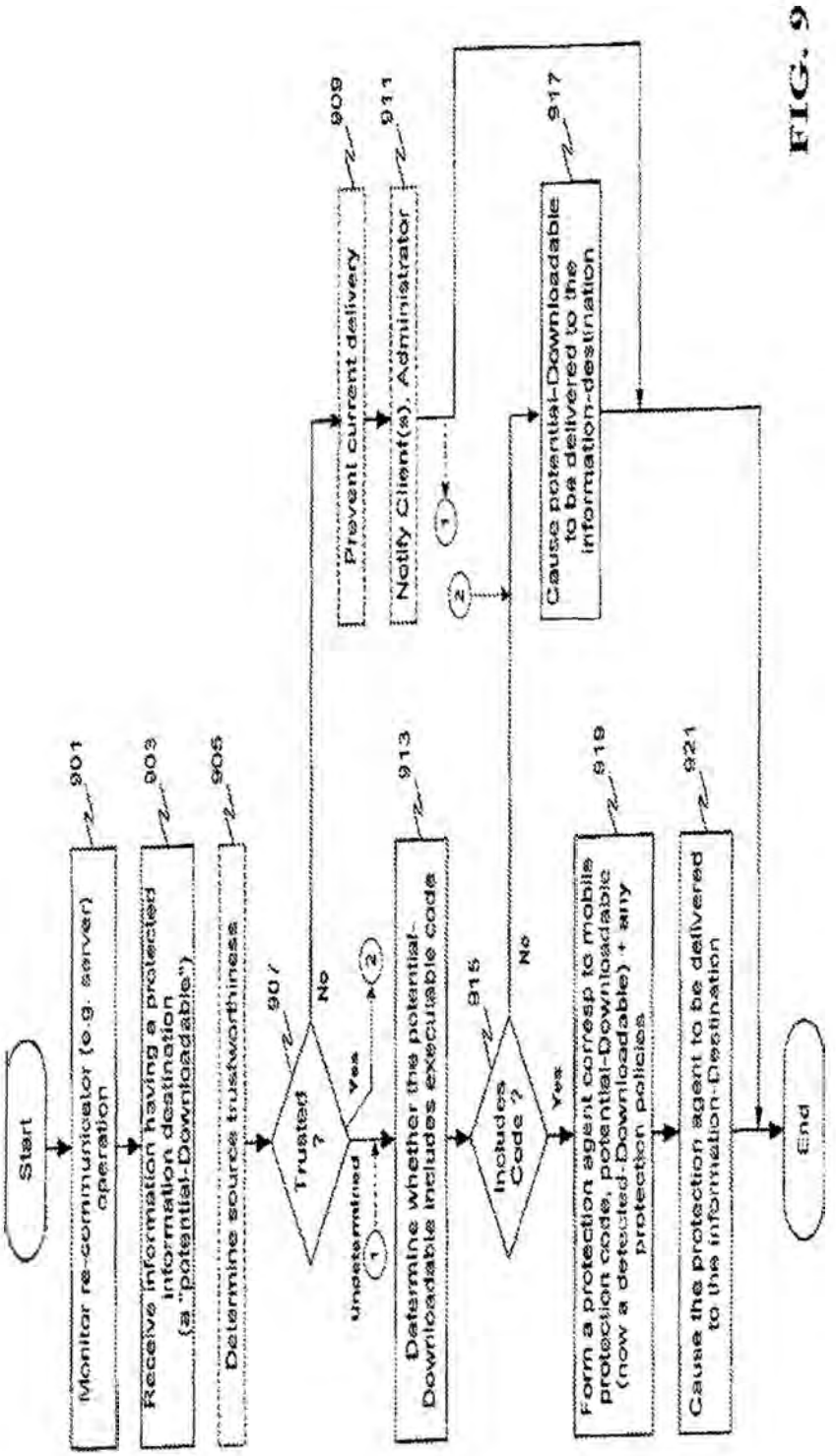

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
Trusted ?
907
No
Yes
Untrusted
Prevent current delivery
909
Notify Client(s), Administrator
911
Determine whether the potential-Downloadable includes executable code
913
Includes Code ?
915
No
Yes
Cause potential-Downloadable to be delivered to the information-destination
917
Form a protection agent corresp to mobile protection code, potential-Downloadable (now a detected-Downloadable) + any protection policies
919
Cause the protection agent to be delivered to the information-Destination
921
End

FIG. 9

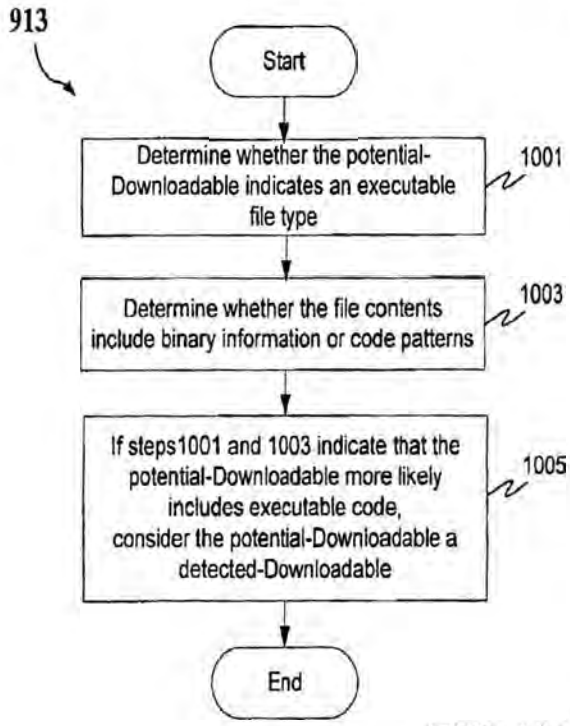


FIG. 10A

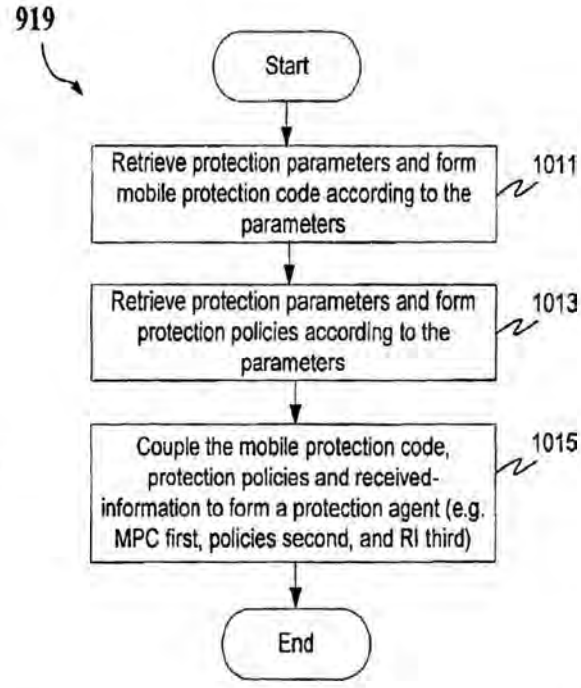


FIG. 10B

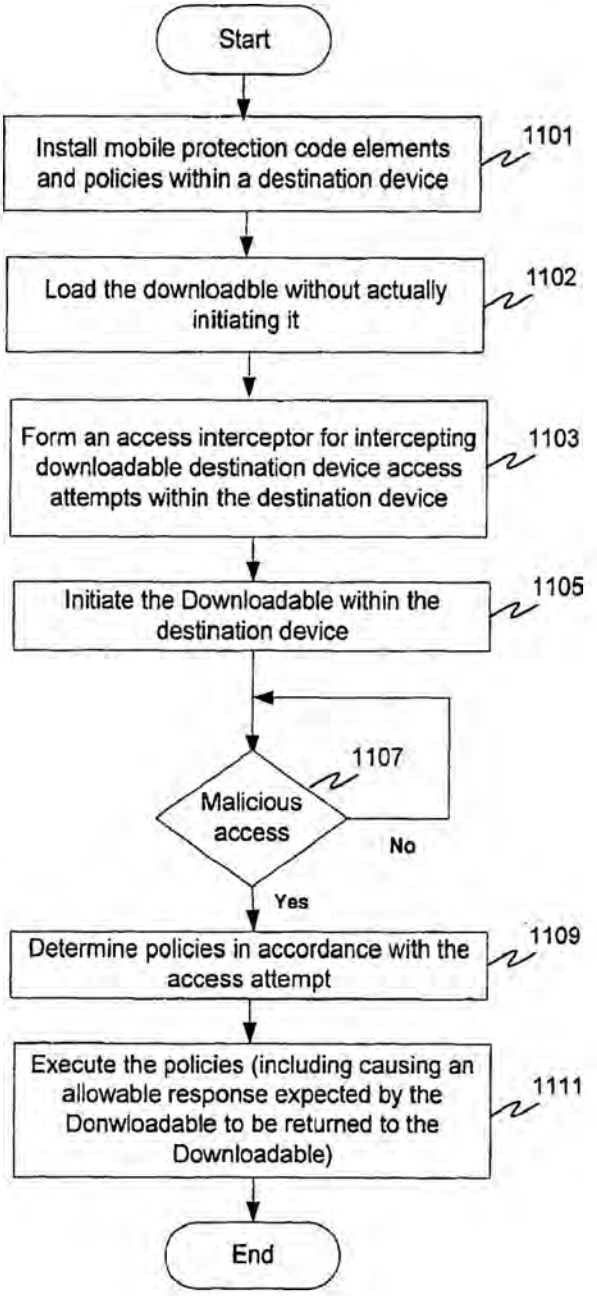

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable within the destination device
1105
1107
Malicious access
No
Yes
Determine policies in accordance with the access attempt
1109
Execute the policies (including causing an allowable response expected by the Donwloadable to be returned to the Downloadable)
1111
End

FIG. 11

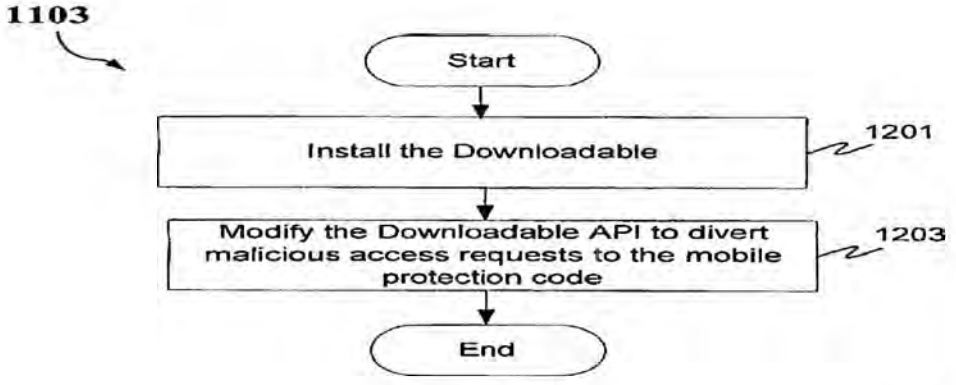


FIG. 12a

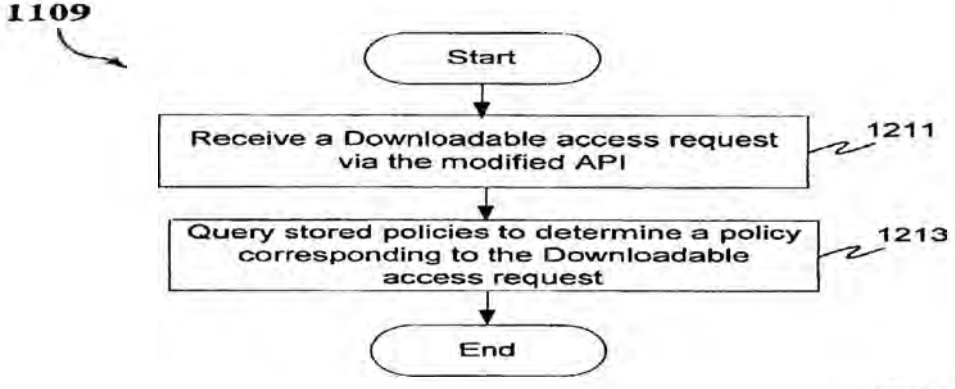


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application is a division of U.S. patent application Ser. No. 14/619,363, filed Feb. 11, 2015 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application Ser. No. 14/155,835, filed Jan. 15, 2014 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application Ser. No. 13/290,708, filed Nov. 7, 2011 by inventors Yigal Mordechai Edery, et al., entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of U.S. patent application Ser. No. 12/471,942, filed May 26, 2009 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 8,079,086, entitled "Malicious Mobile Code Runtime Monitoring System and Methods," which is a continuation of assignee's U.S. patent application Ser. No. 11/370,114, filed Mar. 7, 2006 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 7,613,926, entitled "Method and System for Protecting a Computer and a Network from Hostile Downloadables," which is a continuation of assignee's U.S. patent application Ser. No. 09/861,229, filed on May 17, 2001 by inventors Yigal Mordechai Edery, et al., now U.S. Pat. No. 7,058,822, entitled "Malicious Mobile Code Runtime Monitoring System And Methods," all of which are hereby incorporated by reference. U.S. patent application Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, claims benefit of provisional U.S. patent 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. patent application Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, is also a Continuation-In-Part of assignee's 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 U.S. patent application Ser. No. 08/964,388, filed on Nov. 6, 1997 by inventor Shlomo Touboul, 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, which application claims the benefit of provisional U.S. application Ser. No. 60/030,639, filed Nov. 8, 1996 by inventors Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables." U.S. Ser. No. 09/861,229, now U.S. Pat. No. 7,058,822, is also a Continuation-In-Part of assignee's 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, filed Jan. 29, 1997 by inventor Shlomo Touboul, now U.S. Pat. No. 6,167,520, entitled "System and Method For Protecting a Client From Hostile Downloadables" which claims the benefit of U.S. provisional application No. 60/030,639, filed on Nov. 8, 1996 by inventor Shlomo Touboul, entitled "System and Method For Protecting a Computer From Hostile Downloadables."

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

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

FIG. **1***c* 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. **6***a* is a block diagram illustrating protection engine parameters according to an embodiment of the invention;

FIG. **6***b* 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. **7***a* is a flow diagram illustrating a sandbox protection system operating within a destination system, according to an embodiment of the invention;

FIG. **7***b* is a block diagram illustrating memory allocation usable in conjunction with the protection system of FIG. **7***a*, 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. **10***a* 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. **10***b* 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. **12***a* is a flowchart illustrating a method for forming a Downloadable access interceptor according to an embodiment of the invention; and

FIG. **12***b* 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. **1***a* through **1***c* illustrate a computer network system **100** according to an embodiment of the invention. FIG. **1***a* broadly illustrates system **100**, while FIGS. **1***b* and **12** of **1***c* illustrate exemplary protectable subsystem implementations corresponding with system **104** or **106** of FIG. **1***a*.

Beginning with FIG. **1***a*, 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 modern, 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. **1***a* 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. **1***b* and **1***c* 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** band firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements **11**) 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, 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-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, OSL, infrared or other suitable transceiver, etc. for providing inter-device commu-

nication 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 **110** 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 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 NM 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 (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 me 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 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 **20** (e.g. see FIG. **6***b*). 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 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. (Anyone 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**, inflator **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 as 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 inflator **504**. File inflator **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, inflator **504** returns such information to parser **502**. File inflator **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. **6***b*). 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 **643***b* of FIG. **6***b*. (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 (**643***a* of FIG. **6***b*). 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 **643***c*, 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. **6***a* 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. **6***b* 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 concatination 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. **7***a*, **7***b* and **8** illustrate a "sandbox protection" system, as operable within a receiving destination-device, according to an embodiment of the invention.

Beginning with FIG. **7***a*, 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, PI. 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 IA T can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IA T 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.

Turning to FIG. **8** with reference to FIG. **7***b*, 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 PI) 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 as/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 **20** 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. **10***a* 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. **10***a*, 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. **10***b* 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. **12***a* example, the MPC can provide for intercepting Downloadable access attempts by a Down-

loadable 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 ill, 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. **12***b* 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 system for reviewing an operating system call issued by a downloadable, comprising:

at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:

an operating system probe associated with an operating system function for intercepting an operating system call being issued by a downloadable to an operating system and associated with the operating system function;

a runtime environment monitor for comparing the operating system call against a predetermined security policy including multiple security rules to determine if execution of the operating system call violates one or more of the multiple security rules before allowing the operating system to process the operating system call and for forwarding a message to a response engine when the comparison by the runtime environment monitor indicates a violation of one or more of the multiple security rules;

a response engine for compiling each rule violation indicated in the messages forwarded by the runtime environment monitor, for blocking execution of operating system calls that are forbidden according to the security policy when execution of the operating system calls would result in a violation of a predetermined combination of multiple security rules of the predetermined security policy and for allowing execution of operating system calls that are permitted according to the security policy;

a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;

a request broker for receiving a notification message from the downloadable engine regarding the extension call;

a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine and intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function;

an event router for receiving the notification message from the request broker regarding the extension call and an event message from one of the file system probe and the network system probe regarding the operating system call;

the runtime environment monitor for receiving the notification message and the event message from the event router and comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and

the response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

2. The system of claim **1**, wherein the downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.

3. A system for reviewing an operating system call issued by a downloadable, comprising:

at least one processor for accessing elements stored in at least one memory associated with the at least one processor and for executing instructions associated with the elements, the elements including:

a downloadable engine for intercepting a request message being issued by a downloadable to an operating system, wherein the request message includes an extension call;

a request broker for receiving a notification message from the downloadable engine regarding the extension call;

a file system probe and a network system probe each being associated with an operating system function for receiving the request message from the downloadable engine, intercepting an operating system call being issued by the downloadable to an operating system and associated with the operating system function and providing an event message regarding the operating system call;

a runtime environment monitor for receiving the notification message and the event message and comparing the extension call and the operating system call against a predetermined security policy before allowing the operating system to process the extension call and the operating system call; and

a response engine for receiving a violation message from the runtime environment monitor when one of the extension call and the operating system call violate one or more rules of the predetermined security policy and blocking extension calls and operating system calls that are forbidden according to the predetermined security policy, and for allowing extension calls and operating system calls that are permitted according to the predetermined security policy.

4. The system of claim **3**, wherein the Downloadable is one of a Java component, an ActiveX control, executable code, or interpretable code.

5. The system of claim **3**, wherein the response engine stores results of the comparison in an event log.

6. The system of claim 3, wherein the response engine informs the user when the security policy has been violated.

7. The system of claim 3, wherein the response engine stores information on the Downloadable in a suspicious Downloadable database when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy.

8. The system of claim 3, wherein the response engine discards the Downloadable when it includes extension calls or operating system calls that are forbidden according to the predetermined security policy.

* * * * *

CERTIFICATE OF SERVICE AND FILING

I certify that on August 23, 2021, I electronically filed the foregoing **OPENING BRIEF OF FINJAN LLC** using the Court's CM/ECF filing system. Counsel for appellee were electronically served by and through the Court's CM/ECF filing system per Fed. R. App. P. 25 and Fed. Cir. R. 25(e).

/s/ Juanita R. Brooks
Juanita R. Brooks

CERTIFICATE OF COMPLIANCE

Pursuant to Federal Rule of Appellate Procedure 32(g) and Federal Circuit Rule 32(b)(3), the undersigned counsel hereby certifies that:

1. This brief complies with the type-volume limitations of Federal Circuit Rule 32(b)(1). This brief contains 10,232 words, excluding the parts of the brief exempted by Federal Rule of Appellate Procedure 32(f) and Federal Circuit Rule 32(b)(2).

2. This brief complies with the typeface requirements of Federal Rule of Appellate Procedure 32(a)(5) and the type-style requirements of Federal Rule of Appellate Procedure 32(a)(6). This brief has been prepared using Microsoft Word 2016 in a proportionally spaced typeface: Cambria, font size 14 point.

Dated: August 23, 2021

/s/ Juanita R. Brooks
Juanita R. Brooks