EX 1006

11324 U.S. PTO
092704

APPLICANT DOCKET NO. RJ-025

17513 U.S. PTO
60/613909
092704

IN THE U.S. PATENT AND TRADEMARK OFFICE
Provisional Application Cover Sheet

Mail Stop Provisional Patent Application
Commissioner for Patents
P. O. Box 1450
Alexandria, VA 22313-1450

Sir:

This is a request for filing a PROVISIONAL APPLICATION under 37 CFR 1.53(c).

INVENTOR(s)/APPLICANT(s)

Last Name	First Name, MI	Residence (City and Either State or Foreign Country)
Roskind	James A.	Redwood City, CA
Emigh	Aaron T.	Incline Village, NV

TITLE OF THE INVENTION

SYSTEM PRIVACY AND SECURITY

CORRESPONDENCE ADDRESS

James A. Roskind
920 Governors Bay Drive
Redwood City CA, 94065
US

ENCLOSED APPLICATION PARTS (check all that apply)

(X) Specification *Number of Pages* 35
(X) Drawing(s) *Number of Pages* 19
( ) Power of Attorney
( ) Additional inventors are being named on separately numbered sheets attached hereto.

METHOD OF PAYMENT

(X) $80.00 Small Entity payment by credit card. Form PTO-2038 is attached.

(X) At any time during the pendency of this application, please charge any fees required or credit any overpayment to the credit card information provided on Form PTO-2038.

Respectfully submitted,

James A. Roskind
Applicant

Date: September 27, 2004

Telephone No.: 650-595-8448

"Express Mail" label no. EO 905 255 608 US

Date of Deposit: September 27, 2004

I hereby certify that this is being deposited with the United States Postal Service 'Express Mail Post Office to Addressee' service under 37 CFR 1.10 on the date indicated above and is addressed to the Mail Stop Provisional Application, Commissioner for Patents, P O Box 1450, Alexandria, VA 22313-1450.

By
James A. Roskind

Applicant Docket No. RJ-025

APPLICATION FOR UNITED STATES PATENT

Contagion Isolation and Inoculation

By Inventors:

James A. Roskind
920 Governors Bay Drive, Redwood City CA 94065
A Citizen of the United States of America

Aaron T. Emigh
762 Judith Court, Incline Village NV 89451
A Citizen of the United States of America

Telephone 650-595-8448

CONTAGION ISOLATION AND INOCULATION

BRIEF DESCRIPTION OF THE DRAWINGS

Various embodiments of the invention are disclosed in the following detailed description and the accompanying drawings.

FIG 1 is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments.

FIG 2 is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments.

FIG 3 is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments.

FIG 4 is a flow diagram of a method for acquiring an update according to some embodiments.

FIG 5 is a flow diagram of a method for distributing updates according to some embodiments.

FIG 6 is a flow diagram of a method for establishing a redistribution policy according to some embodiments.

FIG 7 is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG 8 is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments.

FIG 9 is a flow diagram of a method for redistributing content such as updates according to some embodiments.

FIG 10 is a flow diagram of a method for identifying and remedying problems according to some embodiments.

FIG 11 is a flow diagram of a method for identifying vulnerabilities according to some embodiments.

FIG 12 is a flow diagram of a method for identifying infestations according to some embodiments.

FIG 13 is a flow diagram of a method for monitoring addressed computers for infestations according to some embodiments.

FIG 14 is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG 15 is a flow diagram of a method for quarantining a computer according to some embodiments.

FIG 16 is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments.

FIG 17 is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments.

FIG 18 is a flow diagram of a method for categorizing messages according to some embodiments.

FIG 19 is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FORTINET EX1006 Page 5

DETAILED DESCRIPTION

The invention can be implemented in numerous ways, including as a process, an apparatus, a system, a composition of matter, a computer readable medium such as a computer readable storage medium or a computer network wherein program instructions are sent over optical or electronic communication links. In this specification, these implementations, or any other form that the invention may take, may be referred to as techniques. In general, the order of the steps of disclosed processes may be altered within the scope of the invention.

A detailed description of one or more embodiments of the invention is provided below along with accompanying figures that illustrate the principles of the invention. The invention is described in connection with such embodiments, but the invention is not limited to any embodiment. The scope of the invention is not limited by these embodiments and the invention encompasses numerous alternatives, modifications and equivalents. Numerous specific details are set forth in the following description in order to provide a thorough understanding of the invention. These details are provided for the purpose of example and the invention may be practiced without some or all of these specific details. For the purpose of clarity, technical material that is known in the technical fields related to the invention has not been described in detail so that the invention is not unnecessarily obscured.

Contagion refers herein to software programs that are designed to perform undesirable actions from the perspective of the (authorized) user of a computer, and at the same time perform desired actions from the perspective of the creator or controller of such contagion. Examples of contagion include computer worms, viruses, and Trojan horses. Anti-contagion software refers herein to software that prevents, impedes or remediates contagion, such as Norton Antivirus from Symantec, or VirusScan from McAfee, which identifies and/or removes contagion from a user's computer. Another example of anti-contagion software is software that repairs a security vulnerability through which contagion may spread. Many anti-contagion methods rely on contagion descriptions, such as virus signatures, to identify the contagion, and must be updated regularly to be effective.

FIG 1 is a flow diagram of a method for reducing an anti-contagion update interval during periods of suspicious network activity according to some embodiments. An update interval is the period of time between acquisition of updates, such as security updates, virus signature file updates, or virus scanner updates. In this example, network traffic statistics may be monitored (101). An example of traffic statistics is latency experienced by email, such as average latency, or median latency. Another example of traffic statistics is the rate of unacceptable attempted TCP/IP traffic, such as attempts to connect to blocked or restricted ports, or insufficiently authenticated (or authorized) connections to ports, or connections that provided invalid or illegal content such as extraordinarily long strings, or specific content designed to be malicious. Recent

activity may be evaluated (102) to see if it is anomalous, such as median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, or rates of unacceptable attempted TCP/IP traffic beyond some threshold, for example greater than two attempts per minutes over a one hour period. Another example of a threshold for anomalous activity is a comparison with recent historical statistics, such as unacceptable attempted TCP/IP traffic at a rate greater than ten times the expected rate over a 30 minute period. The expected rate may for example be an average over a recent historic period, such as an average rate over the last week of connectivity. If activity is not deemed ahomalous (103), then in this example, processing continues with no change (104) in the next update interval. If the activity is deemed anomalous (103), then in this example, the next update interval is reduced, for example reduced to some proportion of the planned update interval, for example reduced to 25% of the previous or default update interval.

FIG 2 is a flow diagram of a method for cooperatively scheduling an anti-contagion update interval according to some embodiments. In this example, the planned time for contacting an anti-contagion source may arrive (201). An old version identifier may be retrieved (202), such as a checksum or hash for a file, or a version number for software stored in a file system or registry, or a version number for an anti-virus signature file stored in a file system or registry. An update may be requested from an anti-contagion source (203), for example as is discussed in conjunction with FIG 3, which may include updating or concluding

that an update is not needed. A time for next contacting an anti-contagion source may be decided upon (204). In some embodiments, the scheduled time may be made sooner when the source anticipates the impending release of a fully tested update, such as scheduling a next contact time in a few hours, or a few days. If the current computer's version of anti-contagion software contains resolutions to all known problems and otherwise anomalous traffic patterns then, in this example, the update interval may be made longer, such as the full default interval for a computer, for example one week. In some embodiments, the selection of the next update interval may be dependent on the specific product. For example, a "pro version" may have a short default interval, such as one day, and a "home version" may have a relatively long default interval, such as two weeks or one month.

FIG 3 is a flow diagram of a method for contacting an anti-contagion source and optionally getting an update according to some embodiments. In this example, an old version identifier may be retrieved (202), for example as was discussed in 202 of FIG 2. Contact may be made with an anti-contagion source (302) via a network such as the Internet. The old version identifier may be provided, for example using a transport protocol such as TCP/IP, and a data encapsulation format such as XML, or using a higher level protocol and format such as SOAP. A decision may be made as to whether an update is needed (303). In some embodiments, this update decision may be made by a server, upon review of the old version identifier. In some embodiments, this update

decision may be made by the user's computer. In some embodiments, the update decision may be made by incorporating preferences held on a user's computer, such as a preference to always have the newest updates, or preferences to avoid updates that have been available for less than a threshold length of time (for example, 2 weeks), or preferences that accept an additional update when the source indicates urgency associated with the update. In some embodiments, the update decision may be made based on considerations such as the size of the potential update, for example by avoiding the overhead of small updates, and only performing an update when a threshold amount of change has taken place, or when a threshold length of time has elapsed since an actual update was made. An example of a threshold amount of change taking place is the addition of a prescribed number of viral signatures, such as 2 new signatures. An example of a threshold length of time is 3 weeks. If the update is not needed (303) then the process of contacting the source is complete (304) in this example. If an update is needed (303) then an update is acquired in this example via direct or indirect channels (305), for example as further described in conjunction with FIG 4. After the update has been acquired (305), the contact may be complete (306).

FIG 4 is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an update source may be contacted (401), such as via an Internet connection, for example via TCP/IP. In some embodiments, the source contact may optionally include an offer to be a

FORTINET EX1006 Page 10

redistribution source (402). A redistribution source may be contacted by other computers to acquire an update. An example of an offer to be a redistribution server is described in conjunction with FIG 6. An update or an update identifier may be acquired (403). An example of an update identifier is a version number for an update. Another example of an update identifier is a checksum for an update. Another example of an update identifier is a set of checksums (such as MD5, SHA-1, or CRC-32) for a corresponding set of byte ranges that form the update. Another example of an update identifier is a combination two or more identifiers, such as the identifiers enumerated above. If the update was completed (404) then the user optionally confirms redistribution sourcing status (408) in this example. In some embodiments, confirmation may include asserting that the client is prepared to redistribute copies of the received data to one or more recipients. If the update was not completed (404), then a list of peer or other sources is optionally acquired (405) in this example. In some embodiments, the list may be explicit. In some embodiments, the list may be implicit, for example represented by an iterator or accessor, which may retrieve elements on an as needed basis. In some embodiments, implicit lists may be dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources may contain addresses that may be contacted to acquire an update, such as a list of IP addresses and ports, as well as optionally some authentication information to be used when contacting a listed source. Attempts may be made to acquire the update from a listed source (406), for example as exemplified in FIG 8. If the update is

completed via a listed source (407), then redistribution status is optionally confirmed (408) in this example. If the update is not complete (407) then a reservation is optionally negotiated and/or acquired (409) in this example. In some embodiments, a reservation may include a time and/or a specific source or IP address. In some embodiments, a reservation address may be selected from a collection of source addresses, for example in order to distribute the load across more than one server, or to provide a server that is closer to a recipient. An example of selecting a server close to a recipient is to select a server having relatively lower latency during a download, or relatively lower cost, for example by selecting a source within a recipient's ISP complex. If a reservation time was acquired, then a wait for that reservation time (410) is optionally performed in this example. An attempt to acquire an update or update identifier from the source (403) may be again attempted, and the flow may proceed as described earlier. This flow may continue until an acquisition is complete (408), as also described earlier.

FIG 5 is a flow diagram of a method for distributing updates according to some embodiments. In this example, a source prepares to supply updates to recipients (501), for example by starting up a server, for example an HTTP or FTP server, or a multicast server, such as an RTP multicast server. The server waits for an event (502), such as a contact on a predefined port, or a notification that the server should start a multicast. If a multicast-time event appears (503) then an update is multicast (504) in this example. In an embodiment, a multicast

FORTINET EX1006 Page 12

time event may be generated internally or externally to the server to start a multicast. In some embodiments, a multicast-time event may be automatically generated at predetermined times, such as hourly, weekly, or daily, or on individually specified dates and times. In some embodiments, a multicast-time event may include a specific identifier for the multicast. If an event of the form of an update request arrives (503) then a test for authenticity is optionally performed (505) in this example. An example of determining whether a request is authentic is to check a credential. Examples of credentials include providing a shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor such as a user serial number, or a cryptographically signed authentication, such as credentials or a certificate issued to users that had paid for an update service, for example, by inclusion in the purchased software package. In some embodiments, all requests are assumed to be authentic without the use of any authenticator. If the request was not authentic (505) then, in this example, the server continues to wait for an event (502). If the request was authentic (505) then the server may determine a necessary update (506) in this example. In some embodiments, determination of an update may be made based on the requestor's state, which in some embodiments may be indicated in a received request for update. A redistribution contract may be negotiated (507). In some embodiments, some requestors may agree to redistribute updates to other requestors. In some embodiments, some requestors may decline to redistribute updates. In some embodiments, a source may decline redistribution proposals from a requestor. An example of a

redistribution contract is an agreement not to redistribute. Another example of a redistribution contract is an agreement to redistribute a fixed number of copies, such as five copies. Another example of a redistribution contract is an agreement to redistribute for a fixed period of time, such as one day. Another example of a redistribution contract is an agreement to redistribute within a specified geographic region within a fixed period of time. Another example of a redistribution contract is an agreement to redistribute indefinitely, for example until a replacement update is provided. A redistribution contract may be made with terms that are combinations of the above mentioned exemplary term limitations. An update may be provided directly or indirectly (508). A direct update may for example be provided by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and an appropriate coding, such as uuencode. An indirect update may be provided by directing the requestor to an alternate source, such as a redistribution server, or a server at a different address, or a server at a different time, such as a future time, or a combination of two or more such specifications. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on the redistribution contract, for example a requestor that is willing to redistribute may be updated sooner or more directly. An example of a redistribution contract is a promise by a user to provide redistribution in exchange for purchase price concession, such as a reduced initial purchase price, or a quantity discount price. In some embodiments, a decision to provide direct vs. indirect updates may be based in part on previous agreements, such as

prepayment for expedited service, such as may be included in a more expensive version of a product that utilizes the updates. The server may wait for an event (502). In some embodiments, events may be processed in parallel with previous events, or may be sequentially processed, or some combination, for example up to a threshold number of events in parallel, such as a maximum of 32 events in parallel.

FIG 6 is a flow diagram of a method for establishing a redistribution policy according to some embodiments. In this example, a computer may prepare to offer to be a redistribution source (601). In some embodiments, preparations may include retrieving preferences, such as the number of redistributions that may be done, or the duration of time that the computer is willing to act as a redistribution source. In some embodiments, preferences may be defaults, or may be configured by a user and stored persistently, such as in a registry or file system. A contact address may be proposed to the source and a policy may be negotiated (602). A contact address may for example be a globally routable IP address and a port, or a local IP address (reachable from a LAN behind a firewall) and port that can be accessed by other computers on the LAN, such as computers that shared a common NAT address in their contact with the original source. In some embodiments, negotiation may include proposing a policy. In some embodiments, negotiations may involve a counter proposal in response to a proposal. An example of a counter proposal criterion is to require a minimum duration for redistribution. If terms were not acceptable (603) then, in this

example, the computer does nothing (604). If the terms were acceptable (603) then, in this example, the computer prepares to service redistribution requests (605), for example by establishing a server to respond at the agreed contact address, reserving storage to hold the content for redistribution, etc. In some embodiments, redistribution policy may be a broader contractual matter, where an update is provided more quickly or more directly in exchange for a promise to redistribute. In some embodiments, an auditing mechanism may be used to measure compliance and penalize contract breakers, such as failure to accept connections for redistribution, where penalties may for example include refusal to accept future similar contracts. As another example, a lower purchase or subscription price may be offered for operators of a software product, such as antivirus software, who agree to act as redistribution points. In some embodiments such purchase agreements may be configured into the software, for example by establishing agreements to act as a redistribution point for updates as defaults that may be unmodifiable by a user.

FIG 7 is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a redistribution computer is contacted and confirms its sourcing status (701), for example by accepting a TCP/IP connection on a predefined port, and processing requests for content. The redistribution computer may optionally contact another computer, for example a more original source, and affirm its redistribution sourcing status (702). A more original source may be a source that has provided content in the

past to the redistribution computer, directly or indirectly. In some embodiments, contacting a more original source may demonstrate that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source may allow the redistribution computer to further authenticate the computer that requested the content, for example by getting confirmation from a more original source that a specific computer was authorized to request the content. The requesting computer may be required to provide satisfactory authentication by some agreeable method (703). In some embodiments, authentication includes providing a shared secret to the redistribution computer. In some embodiments, authentication includes presenting an authorization cryptographically signed by a more original source, such as a certificate. In some embodiments, all requests are presumed authentic. If the request is not authentic (704), then nothing further is done (705) in this example. If the request is authentic (704), then the requested update is supplied (706) in this example. The updates may be supplied for example via a transport protocol TCP/IP, or a higher level protocol such as FTP, HTTP, or HTTPS, etc.

FIG 8 is a flow diagram of a method for using peers or other indirect data sources to acquire content such as updates according to some embodiments. In this example, a list of one or more peers or other indirect sources for acquiring updates is obtained (801), for example as was discussed in conjunction with 405 of FIG 4. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect

FORTINET EX1006 Page 17

source may be a computer configured to redistribute content without other use of the content, such as an ISP server established to redistribute content to subscribers. In another example, an indirect source may be a server created by the original source to perform additional distributions. In some embodiments, an obtained list of indirect sources may be implicit, for example by referring to a list using an iterator or accessor construct that can itemize entries in the list when requested, for example using a query to a service when an additional item in the list is required, which may for example dynamically complete the list on an as requested basis. In some embodiments, a list of sources may be processed by retrieving the next address and (optional) authenticator (802). Any order may be used. In some embodiments, an order of processing may correspond to the original list order. In some embodiments, an order of processing may be such that the earlier processed addresses would be expected to have lower latency and higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list may be processed in parallel, and portions (such as byte ranges) of the content may be acquired from different addresses in the list. An address on the list may be contacted to request content (803), such as was illustrated in FIG 7. In one example, a source address may consist of an identifier for a broadcast with optional time(s) for a broadcast, and contact with the source may include receiving the broadcast. In another example, contact with the source may include making connection such as TCP/IP connections with the source. If the acquisition is complete (804), then the processing is complete (805) in this example. If the update was not

completed (804) then additional sources in the list are considered in this example. If there are more sources in the list (806) then, in this example, a next source is retrieved along with an optional authenticator (802) and the processing continues. If there are no more sources (806) then the attempted acquisition from source(s) is unsuccessful (807) in this example.

FIG 9 is a flow diagram of a method for redistributing content such as updates according to some embodiments. In this example, a computer prepares to supply content (or updates) to a requestor/recipient (901), for example by starting up one or more servers, such as an HTTP server, an FTP server, or a multicast server. A server may wait for an event (902), such as a contact on a predefined port, or a notification that the server should be terminated, or a notification that a time for a multicast has arrived. If a service expiration event appears (904) then the supply service is terminated (905) in this example. A service expiration event may be automatically generated, for example after a predetermined duration of service, or after a predetermined number of copies of an update have been redistributed. If a request event arrives (904), then a test for authenticity is performed (906) in this example. Authenticity may be established in the request, for example by providing a shared secret, such as a secret used by all requestors to this server, or a different secret used by each requestor of this service and held in a list of single-use secrets by this server, or by providing a cryptographically signed authentication. In some embodiments, all requests are authentic without the use of any authenticator. If the request was

not authentic (906) then, in this example, the server continues to wait for an event (902) as described earlier. If the request was authentic (906) then, in this example, the server transmits a copy of the content to the requestor (907), for example by allowing an FTP download, or by encapsulating the transmission in an HTTP transfer, for example using XML and an appropriate coding, such as uuencode.

FIG 10 is a flow diagram of a method for identifying and remedying problems according to some embodiments. A problem may be identified as some event or status that may cause direct or potential harm. Exemplary problems include vulnerability or infestation. In this example, a problem such as a vulnerability or infestation is detected (1001). Infestation herein refers to the current execution of contagion. In some embodiments, detection may be made by a router, firewall, or other network component, separately or in concert, using techniques such as those illustrated by FIG 11, 12, and 13. In some embodiments, detection may be made by a computing device, for example a computing device operated by an ISP. After a detection of a problem, a computer or network containing a problem may be quarantined (1002). In some embodiments, quarantining may prevent unauthorized communications from being sent from the quarantined computer, such as efforts to infect, probe, or harass other computers, or efforts to communicate data without authorization, including for example file contents and keyboard entry logs. In some embodiments, no communications may be authorized. In some embodiments,

FORTINET EX1006 Page 20

quarantining may prevent undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIG 14 and 15. In some embodiments, the quarantined computer may be provided access to remediation services (1003). In some embodiments, access to remediation may include allowing connections to servers providing security patches or advisories, for example allowing connections to a vendor's server, or providing virus and worm disinfecting tools, such as focused removal tools, or data updates for previously installed repair tools. In some embodiments, access to remediation may include direct access to vendor sites known to provide remediation assistance, such as to Microsoft's Windows Update service, where security patches may be obtained. In some embodiments, access to vendor sites may be restricted by the quarantine to only allow certain classes of traffic, such as HTTP requests, certain port access, such as port 80 or port 443, certain content requests, such as specific URLs or specific file downloads. In some embodiments, remediation content may be substituted for other requested content, for example as discussed in conjunction with FIG 14.

FIG 11 is a flow diagram of a method for identifying vulnerabilities according to some embodiments. In this example, preparation is made to scan for one or more vulnerabilities (1101), for example by retrieving one or more IP addresses of computers that need to be scanned. For example, an IP address may be provided by a DHCP sever hosted in an ISP network complex, to identify

FORTINET EX1006 Page 21

current subscriber IP addresses. As another example, a subnet range may be scanned, for example by taking an address of the form A.B.C.X, and using some or all values of X in the range 0 to 255 to enumerate distinct IP addresses. A scan of one or more IP addresses may optionally be performed to detect an open port (1102). An example of scanning an IP address is to detect open TCP/IP ports (1102). An open port may be identified by attempting to form a connection, such as a TCP/IP connection to that port. In some embodiments, only open ports on a given computer will provide acknowledgment messages needed to establish a TCP/IP connections, which for example causes a success code to be returned by Unix "connect()," and a closed port will provide no response to the attempt to connect, which for example would cause an error code to be returned by the Unix function call "connect()". If there are no open ports (1103), then the scanning is complete (1106) in this example. If open ports are detected (1103) then those ports are tested for vulnerabilities (1104) in this example. In some embodiments, an open port is assumed (1103) and a test for vulnerability may be performed (1104), for example by communicating a vulnerability test without independently testing to see if one or more requisite ports are open. In some embodiments, a vulnerability test of a port may include sending data that is known to create a discernable response when a vulnerability is present, such as sending a long string known to induce a buffer overflow with a consequential errant response, where the errant response may for example include premature termination of the connection or other erroneous response(s). In some embodiments, a test for a vulnerability is an attempt to exercise a service, such

FORTINET EX1006 Page 22

as a sendmail or an RPC service, in search of a misconfiguration of that service that creates a vulnerability. One example of a misconfiguration of a service is provision of any such service. Another example of a misconfigured service is one where a password should be required for security purposes, but no password is required by the misconfiguration. Another example of a misconfiguration is the use of an easily guessable password, such as the default installation password, when resulting access creates a security vulnerability. Another example of a misconfigured service is support for a service provided by an unauthorized agent, such as a worm or virus. If no vulnerabilities are detected (1105) then scanning is complete (1106) in this example. If vulnerabilities are detected (1105) then an associated IP address is flagged as vulnerable (1107) in this example. In the case of a subscriber to an ISP, an IP address may be translated to a particular subscriber identity, for example by checking in DHCP or static IP allocation tables for the subscriber's identity, and such identity may be appropriately flagged or recorded. In some embodiments, the record of the vulnerability may additionally include the port, and the details of the vulnerability that was detected, such as which buffer overrun was vulnerable, or which service was misconfigured to create a vulnerability.

FIG 12 is a flow diagram of a method for identifying infestations according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, may be used to monitor a computer for infestation. An anomalous traffic volume, port, or IP destination may be detected

FORTINET EX1006 Page 23

(1202), for example by detecting large volumes of outbound traffic, such as email traffic or instant messaging traffic. Another example of anomalous traffic is a TCP/IP connection to a server that is known to control a collection of infested machines, in a way that suggests the connecting machine is controlled by the server, such as connecting and transmitting an assertion that it is controlled or available to comply. Another example of anomalous traffic is connection to a computer that is a current target of a denial of service attack, with a pattern that is consistent with the attack, such as a connection to a targeted port, or the use of a falsified return IP address. In some embodiments, a falsified return address may be detected by a router, such as an ISP router, that is aware that the return address is not on the sourcing subnet, or not associated with the MAC address that has supplied the corresponding packets. If an anomaly is detected (1203) then an apparent infestation is noted (1204) in this example. In some embodiments, a response to infestation may include quarantining of the computer or network containing the computer, for example as discussed in conjunction with FIGs 14, 15, and 16. In some embodiments, a response to infestation may include an attempt to remediate the problem, for example by inoculations, such as discussed in conjunction with FIG 17. If no anomaly is detected (1203) then monitoring is complete (1205) in this example. In some embodiments, monitoring for anomalous traffic may be continuous. In some embodiments, monitoring for anomalous traffic may be periodic. For example, periodic monitoring may consist of monitoring some number of computers, such as 1 computer, for some period of time, such as 2 minutes, and then not

monitoring those computers again for another period of time, such as 1 hour. As another example, periodic monitoring may be randomized, for example by changing the duration of monitoring, pattern of selecting computers to monitor, and/or the unmonitored duration, to be difficult or impossible to predict, for example by being driven by a pseudo random number generator, subject to restrictions of available monitoring computers.

FIG 13 is a flow diagram of a method for monitoring addressed computers for infestations according to some embodiments. In this example monitoring of an addressed computer for infestation begins (1301), for example by retrieving a list of one or more addresses of computers, such as addresses of participating subscribers. In some embodiments, this list may be a list of addresses for some or all subscribers for a given ISP. In some embodiments, the list may be a list of IP addresses issued by a DHCP server, such as on a LAN, or a list of IP addresses observed by a router, such as a router on a LAN, or a complete list of IP addresses in a given subnet. An addressed computer identified in a list may be queried for a cleanliness assertion (1302), for example by contacting a trusted computing base within a computer, and requesting an authenticated infestation scan by trusted software. An example of a trusted computing base within an operating system is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and operating system. In

FORTINET EX1006 Page 25

some embodiments, trusted code bases may digitally sign assertions about the cleanliness and state of their computers. In some embodiments, the query for cleanliness (1302) may be responded to by anti-contagion software, such as antivirus software, with assertions about the currency of a scan, such as the last time a scan was performed, or the version associated with the current anti-contagion software or definition file in use, where a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with details of its patch level, where a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (1303) then monitoring may be complete (1305) in this example. If a cleanliness assertion is not provided (1303) then an infestation or vulnerability is presumed (1304) in this example.

FIG 14 is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (1401), such as an ISP deciding to quarantine a subscriber, or a firewall or router for a LAN deciding to quarantine a local computer. In some embodiments, a decision to quarantine may be made based on apparent infestations or apparent vulnerabilities, for example as illustrated in conjunction with FIG 11, 12, or 13. In some embodiments a decision to quarantine may be based on the accumulation of evidence above a threshold, such as a threshold of one such piece of evidence, or a threshold of two such pieces of evidence. In some embodiments, a decision to quarantine may be

made based on configuration rules, for example all new computers must be quarantined until they have contacted their OS vendor and been updated, or by user specification, such as a user requesting a quarantine for a specific computer. In some embodiments, requests for a quarantine may be made on an individual computer via settings, such as control panel in Microsoft Windows, or through a user interface to a firewall or router, for example by specifying an IP address, name, or MAC address to quarantine, or a list of addresses that are not to be quarantined (where other addresses may be quarantined). During a quarantine period outbound traffic may be detected (1402), for example by a device such as a router, or by a software network stack on the quarantined computer, or by an upstream device, such as a device at an ISP. Outbound traffic may be tested to see if it is associated with a remediation request (1403), such as contact with an appropriate anti-contagion software distributor, or contact with a security patch update provider, such as Microsoft Windows Update, or contact with a verified remediation site, including for example an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (1403), then the traffic may be routed or forwarded to its destination (1404) in this example. If outbound traffic is not associated with a remediation request (1403), then, in this example, it is tested to see if it is an approved service request (1405), such as a request for a web page. If the request was for an approved service (1405) then, in this example, the request is re-routed to a special quarantine server (1407), such as an ISP web server that provides information and links to assist in remediation. Re-routing of request traffic may be

accomplished in numerous ways, for example by providing an alternate DNS service that consistently returns a fixed IP address of a quarantine server in response to all DNS requests. Another example of re-routing a web page requests is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine sever that provides remediation information. If the request was not for an approved service (1406), then the outbound traffic is discarded or responded to with in indication that the desired traffic destination is unreachable (1406) in this example.

FIG 15 is a flow diagram of a method for quarantining a computer according to some embodiments. In this example, a decision has been made to quarantine a computer (1501), for example as described in conjunction with 1401 Fig 14. Inbound traffic originally destined for a quarantined computer may be detected (1502), for example by a network device such as by a router. An example of detecting inbound traffic destined for a quarantined computer is to have a list of quarantined IP addresses, and filter or provide additional processing on traffic destined for an address on the list. Detected inbound traffic may be tested to see if it is remediation related (1503), such as a response to a remediation request. Direct responses to remediation requests may for example be identified using stateful packet inspection firewall techniques known to those skilled in the art. Traffic from approved sites such as anti-contagion vendors, or security update providers, or internal ISP sites may also be deemed remediation related. If traffic is remediation related (1503) then it is forwarded to the

quarantined computer (1504) in this example. If traffic is not remediation related (1503) then, in this example, it is discarded or responded to with a message indicating that the destination is unreachable (1507).

In some embodiments, a configuration based quarantine during an installation of an operating system, such as Microsoft Windows, may be automatically setup, for example by a default installation setting up restrictions in a firewall installed with the OS or by a default installation setting defaults for use when network connectivity is enabled, that establishes a quarantine, for example restricting internet connections, for example allowing internet connections only to a security update site such as Microsoft Windows Update. In some embodiments, settings that may establish a quarantine, such as default settings in an operating system installation, may be altered programmatically to remove the quarantine in response to a remediation site, such as a security update service such as Microsoft Windows Update, asserting that there are no other security updates pending. In some embodiments, configuration settings that may establish a quarantine may be altered programmatically to enable a quarantine in response to the presence of one or more security updates, such as provided by an operating system update service such as Microsoft Windows Update, that are available for installation on a computer.

FIG 16 is a flow diagram of a method for a quarantine server to respond to requests according to some embodiments. In this example, a quarantine server is started (1601), for example by starting a web server, or DNS server, or proxy

server, or ftp server, or some combination server. The server waits for a request (1602), for example by listening on a port and waiting for a connection. In some embodiments, devices such as routers may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in 1407 in conjunction with FIG 14. If a received connection is a web server request (1603), such as an HTTP request, then the server responds with a quarantine notification page (1604) in this example. An example of a quarantine notification page is a web page that provides notification that the computer is quarantined, or provides links to remediation sites appropriate to the quarantine, such as a link to a site that provides anti-contagion software for removing a virus that the quarantined computer is believed to contain. The links on the quarantine notification page may be examples of HTTP addresses that are for use in remediation. Another example of a quarantine notification page is a page that redirects to a quarantine notification page. If the request is not a web server request (1603), then it is tested to see if it is a DNS inquiry (1605) in this example. A DNS inquiry may for example appear on port 53 using TCP/IP or UDP. If the request is not a DNS inquiry (1605) then it may be discarded or responded to as "unreachable" (1606) in this example. If the request is a DNS inquiry (1605) then, in this example, the inquiry may be tested to see if it is a DNS request for a remediation host name (1608). A remediation host name may include a host name as appropriate to the reason for a quarantine, such as a host providing anti-contagion software, or a host providing security updates, or a host internal to (or partners with) an ISP. If the DNS inquiry was for a

remediation host name (1608) then the request is proxied to an external DNS service provider (1609) in this example. A proxy to a DNS service (1609) may return a stored or cached response, such as the actual IP address of the host name, or it may contact an external DNS service to relay back the proposed IP address. If the DNS inquiry was not for a remediation host name (1608) then an IP address for a quarantine server is provided (1607) in this example. In some embodiments, when a redirected IP address is provided, such as that of a quarantine server, the DNS response may include a short time to live, for example 0 seconds. In some embodiments, the server(s) in this example may continue to wait for a service request (1602) in parallel with the processing of the requests as described above, or may wait for a service request again after disposition of the previous request as described above.

FIG 17 is a flow diagram of a method for scanning backdoors and inoculating according to some embodiments. In this example, preparations are made for scanning a computer for a backdoor inoculation path (1701), for example by identifying the computer to be scanned, and enumerating or retrieving a list of one or more backdoors. In one example, an IP address to be scanned may be selected by an ISP from a list of subscribers. In one example, a list of backdoors may include preloaded software approved by the computer owner for access and inoculation purposes, such as an ISP provided utility that is installed by the subscriber. In some embodiments, an ISP defined backdoor may have an authentication protocol to preclude control by unauthorized parties, such

as restrictions on IP address(es) that may contact the backdoor, or shared secrets such as a password required by the sender, or cryptographic authentication such as client authentication on an SSL connection. As another example, a list of backdoors may include virus or worm installed backdoors, such as those installed by a SubSeven virus or Back Orifice. As another example, backdoors may include exploitable security vulnerabilities, such as a weakness in a web server, or a weakness in an RPC service. A scan may optionally be conducted to detect open TCP/IP backdoor ports (1702) similarly to the scan discussed in conjunction with FIG 11. If none of the backdoors are open (1703), for example there is no response on any of the ports, then the scan is complete (1706) in this example. If backdoor ports are open (1703) then they are tested for usability (1704) in this example. In some embodiments, some backdoor ports are assumed open (1703) and they may be tested for usability (1704) without an independent test to see if the port is open. An example of a usability test is no test at all, with the conclusion that a specific port is usable any time it is open. Another example of a usability test is based on the receipt of confirmation of a functional installation of ISP provided backdoor software, such as a confirmation that may be provided by the installed software. Another example of a usability test involves the use of a less invasive test of vulnerabilities such a buffer overrun exploit that does not have a permanent impact but provides observable validation (such as a dropped connection). If a backdoors is usable (1705) then the computer is inoculated via a backdoor (1707) in this example. One example of such an inoculation is the remotely controlled download of one or more

remedy files into one or more directories, such as an operating system directory, such as the "\windows\system" directory in Microsoft Windows 2000. Another example of an inoculation is the download of an executable file to the computer, followed by the execution of that file. Another example of an inoculation is the download of a data file, such as an anti-contagion signature file, that is interpreted or processed by another executable already on the computer. Another example of an inoculation is the deletion of one or more files from the computer. Another example of an inoculation is the editing of one or more registry entries, or of one or more files, such as deleting, appending or rewriting entries in a registry or file. Another example of inoculation includes a combination of one or more of above techniques, optionally in conjunction with one or more remotely instigated reboot operations. Another example of inoculation includes a combination of one or more of the above techniques, along with a dialog box requesting a user's permission to proceed with the inoculation. Another example of inoculation is the use of a dialog box in conjunction with a timer that causes the inoculation to proceed if no user response is given in a prescribed period of time, such as 1 hour.

FIG 18 is a flow diagram of a method for categorizing messages according to some embodiments. In this example, a message is received (1801), such as an email message or instant message. The purported sender's identity may be extracted (1802), for example by examining the "From:" header line in an email message, or by examining the envelope information conveying an email

FORTINET EX1006 Page 33

message, or by examining the sender's ID in an instant message. A check may be performed to see if the sender has sent any prior messages (1803), for example by scanning an inbox of prior email, or by checking for the sender in a buddy list. If there were no prior messages from the sender (1803) then the message is processed (1804) in this example. Processing the message may for example include presenting the message to the user, and/or recording routing information for comparison and categorization of future messages. If there were prior messages from the sender (1803) then routing information used in prior messages is retrieved (1805) in this example. Routing information may for example be extracted from the header in email. Routing information may for example be obtained during an instant message session by retrieving the IP address of the party being communicated with, for example as is done when an instant message session coordinates a peer-to-peer activity such as a direct file transfer with a remote party. In some embodiments, routing information may be stored implicitly, such as in headers of a previously received message, or explicitly, for example by extracting routing information and storing it in a database or file system. Routing information for the current message may be compared to prior messages (1806), for example by checking that the IP address for the party being communicated with is in a similar subnet, or allocated within the same geographical region, for example the same continent, as a previous IP address for the same party. IP address to geographical location may be estimated by various software applications and services, such as services provided by MaxMind and described on http://www.maxmind.com. In some

embodiments, routing information for an email message may be deduced by examining header information related to mail transfer agents. In some embodiments, similarity in IP addresses for mail transfer agents may be interpreted as implying similar routing. If routing is similar to previous messages from the same sender (1806), then the message is processed as legitimate (1808) in this example. If the routing message purporting to be from a sender is not similar to previous messages from the same sender (1806) then the message may be processed as suspicious (1807) in this example. In some embodiments, processing as suspicious may include contributing to a scoring that labels the message as an undesirable message, such as spam, viral, or phishing. In some embodiments, higher probability suspicious messages may be scrutinized further, or may be restricted to some extent in their impact on the computer. An example of restricting impact of a message may include discarding the message. Another example of restricting impact or further scrutiny of a message may include placing a message in a spam inbox. Another example of a restricting impact of a message may include preventing execution of content contained in the message, and/or requiring additional user interactions to bypass the restriction, for example by having a user agree to ignore warnings which describe the suspicious circumstances.

FIG 19 is a flow diagram of a method for notifying possible sources of infection according to some embodiments. In this example, an infected message is received (1901). An example of an infected message is a message classified

FORTINET EX1006 Page 35

as such by anti-contagion software. Routing information may be extracted from the message (1902). An example of extracting routing information in an email message is to extract header lines inserted by mail transfer agents that propagated the message. Another example of extracting routing information is to acquire an IP address of a party making contact via instant messaging, for example as is provided by AOL's instant message infrastructure for a peer-to-peer file transfer. Routing information from previously received messages may be retrieved (1903). An example of retrieving email routing information from prior messages is scanning all received messages, such as messages in an email inbox, and extracting routing information. Another example of retrieving routing information is to query a database or file system where routing information from previous messages has been gathered. Routing information for the infected message may be compared with the retrieved routing information (1904). If no similarity with previous messages is found (1904) then third party notification is optionally made (1906) in this example. An example of a third party notification is sending notification to the ISP that provides network access to the message source. Another example of notifying a third party is notifying a viral clearinghouse of the apparent source of a viral message. Another example of a notifying a third party is notifying a collaborative filter service that accumulates accusations of viral disseminations. If a similarity is found with previous messages (1904) then a notification may be sent (1905) in this example. In some embodiments, similarity of routing of an infected message with a previous message may include having a previous message from the IP address of the

FORTINET EX1006 Page 36

infected message source. In some embodiments, notification may be sent (1905) to one or more sources of the prior message(s) that best match the infected message, and/or to third parties. As an example, notification may be sent to an email source that had the same originating IP address as the infected message. In some embodiments, a user may be asked to approve notification(s), for example via a pop-up dialog. In some embodiments, notification may be automatically sent without user intervention.

Although the foregoing embodiments have been described in some detail for purposes of clarity of understanding, the invention is not limited to the details provided. There are many alternative ways of implementing the invention. The disclosed embodiments are illustrative and not restrictive.

FORTINET EX1006 Page 37

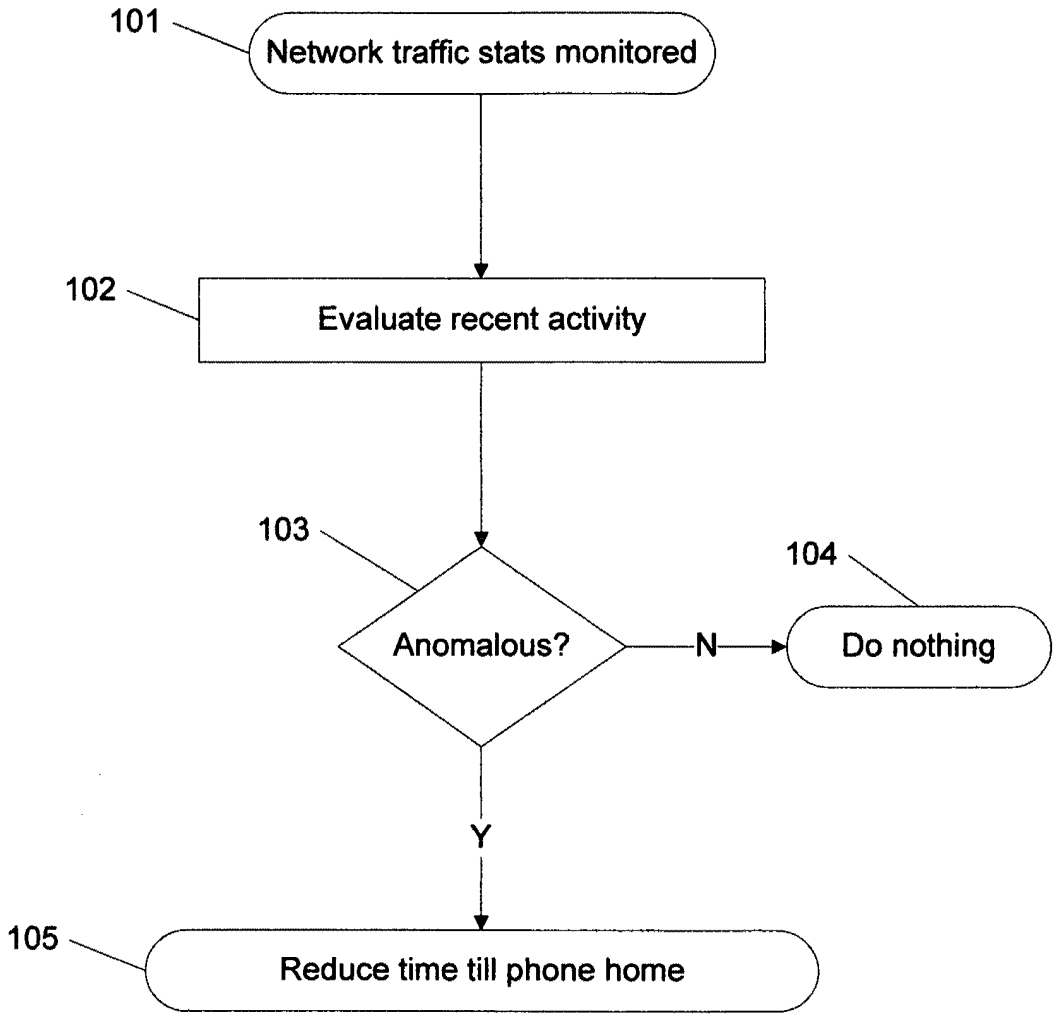
101
Network traffic stats monitored
102
Evaluate recent activity
103
Anomalous?
N
104
Do nothing
Y
105
Reduce time till phone home

Figure 1

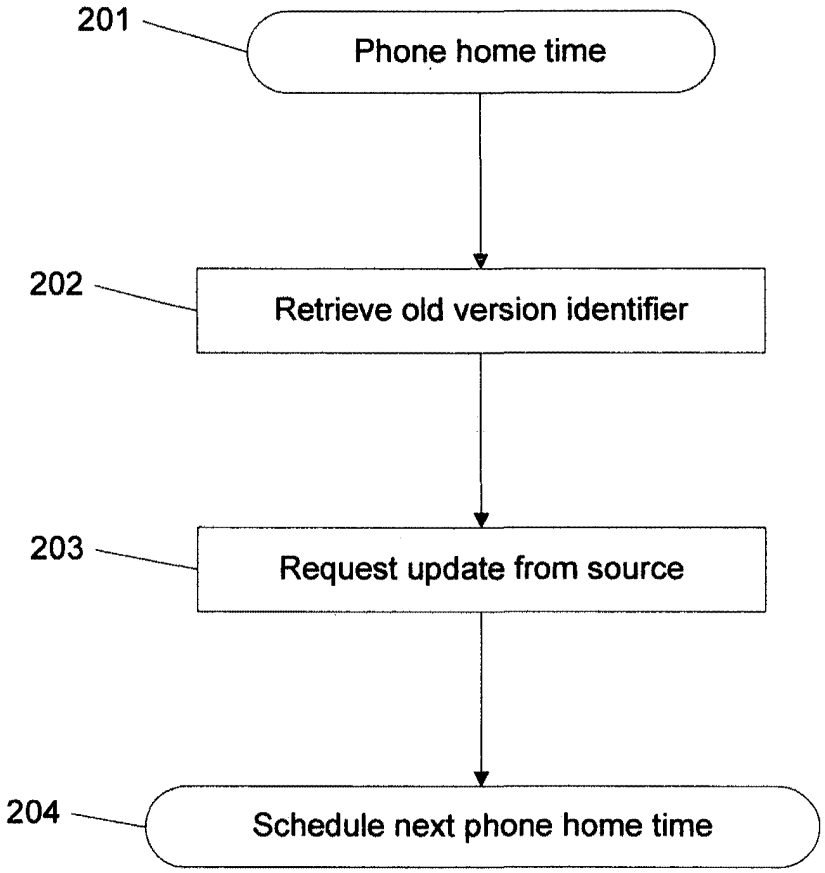
201
Phone home time
202
Retrieve old version identifier
203
Request update from source
204
Schedule next phone home time

Figure 2

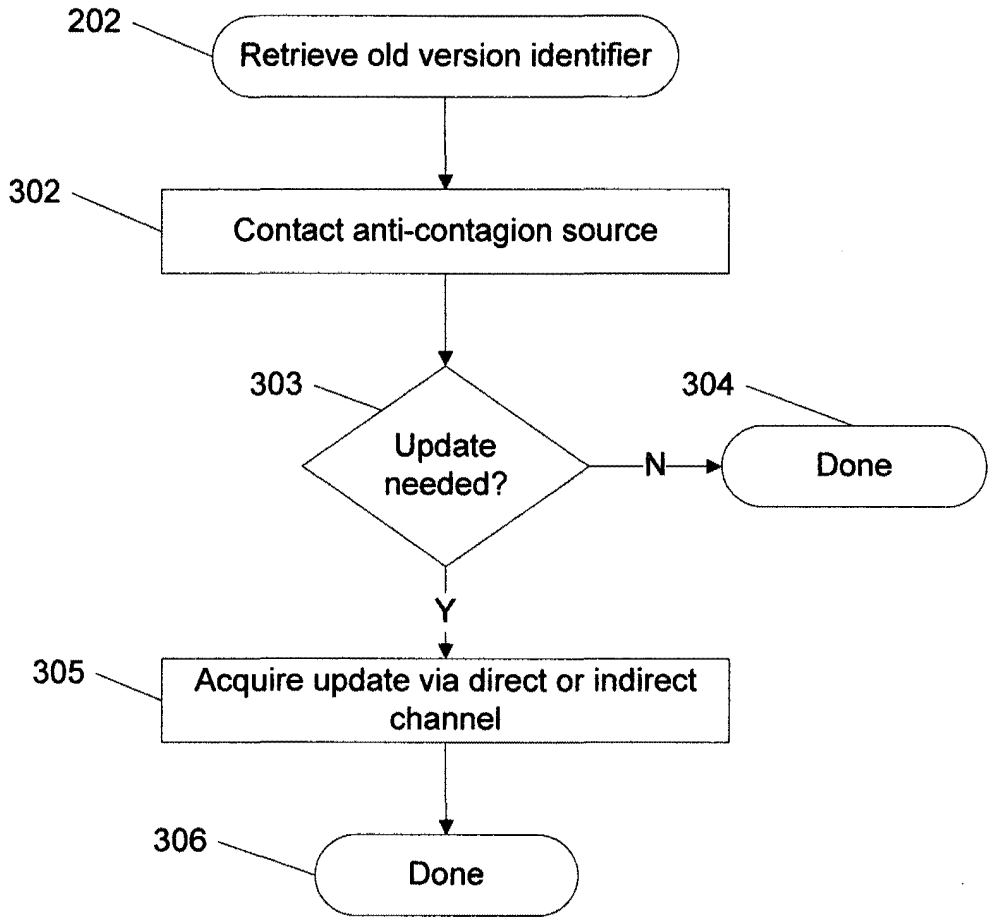
202
Retrieve old version identifier
302
Contact anti-contagion source
303
Update needed?
N
304
Done
Y
305
Acquire update via direct or indirect channel
306
Done

Figure 3

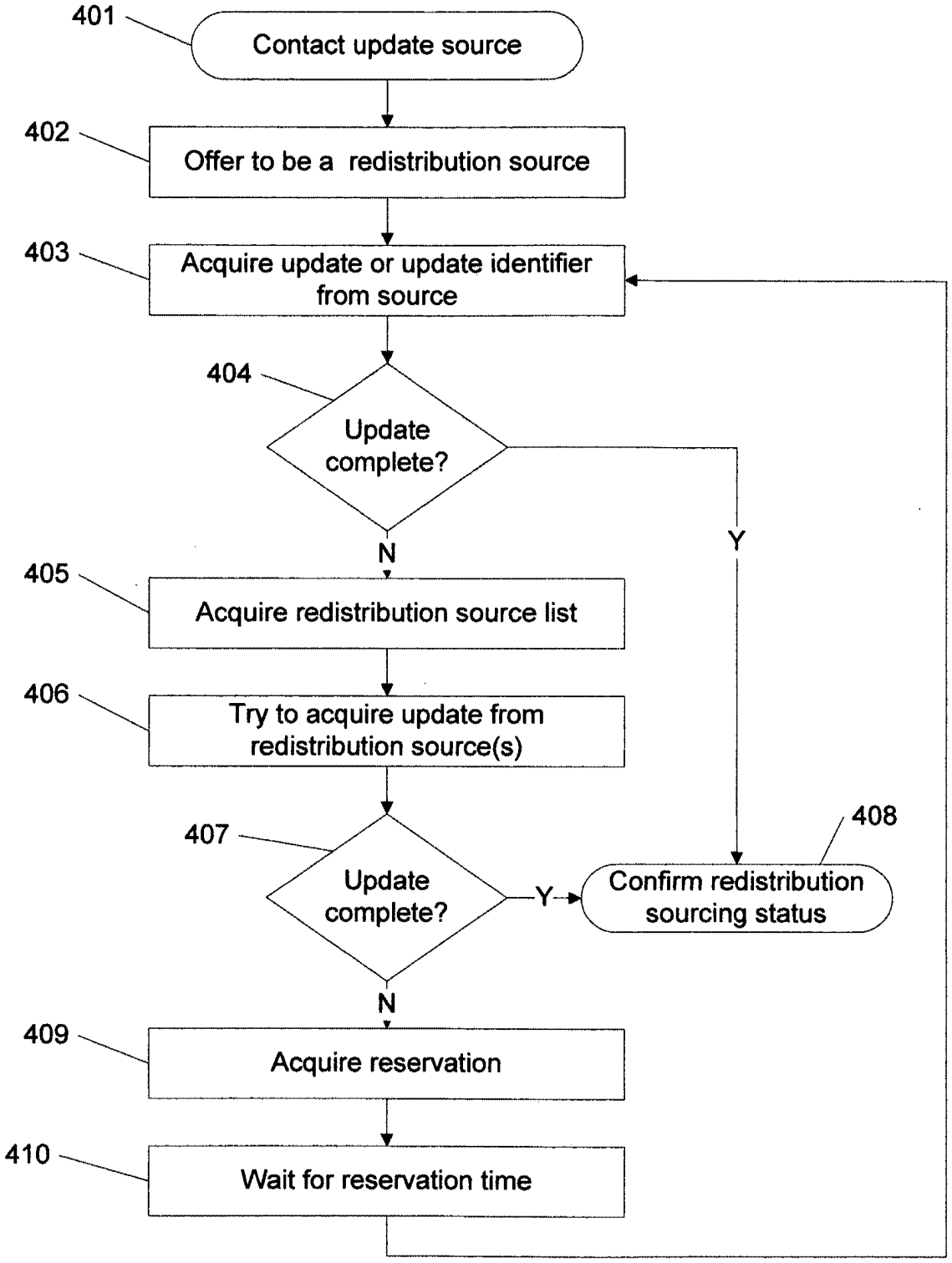
401
Contact update source
402
Offer to be a redistribution source
403
Acquire update or update identifier from source
404
Update complete?
N
Y
405
Acquire redistribution source list
406
Try to acquire update from redistribution source(s)
407
Update complete?
Y
408
Confirm redistribution sourcing status
N
409
Acquire reservation
410
Wait for reservation time

Figure 4

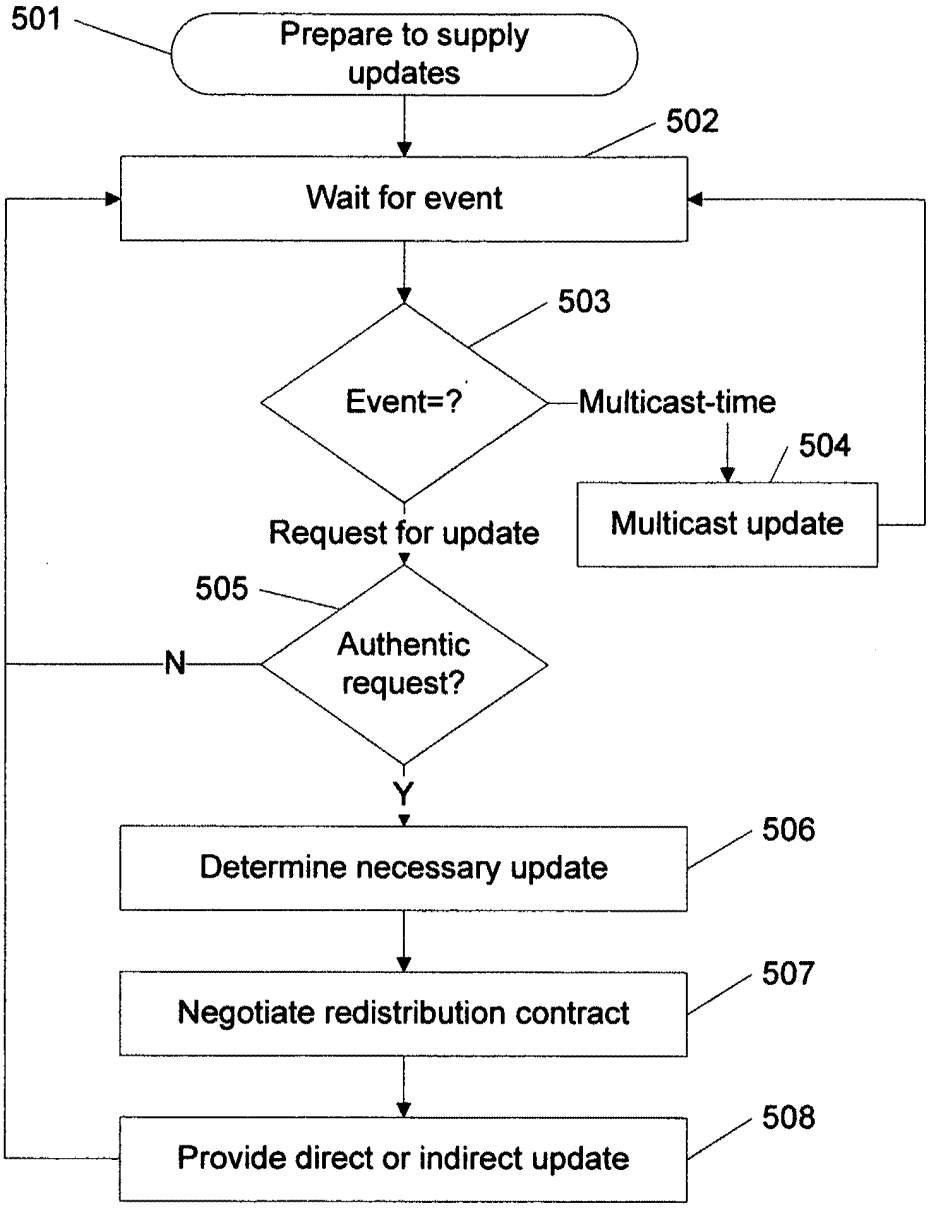
501
Prepare to supply updates
502
Wait for event
503
Event=?
Multicast-time
504
Multicast update
Request for update
505
Authentic request?
N
Y
506
Determine necessary update
507
Negotiate redistribution contract
508
Provide direct or indirect update

Figure 5

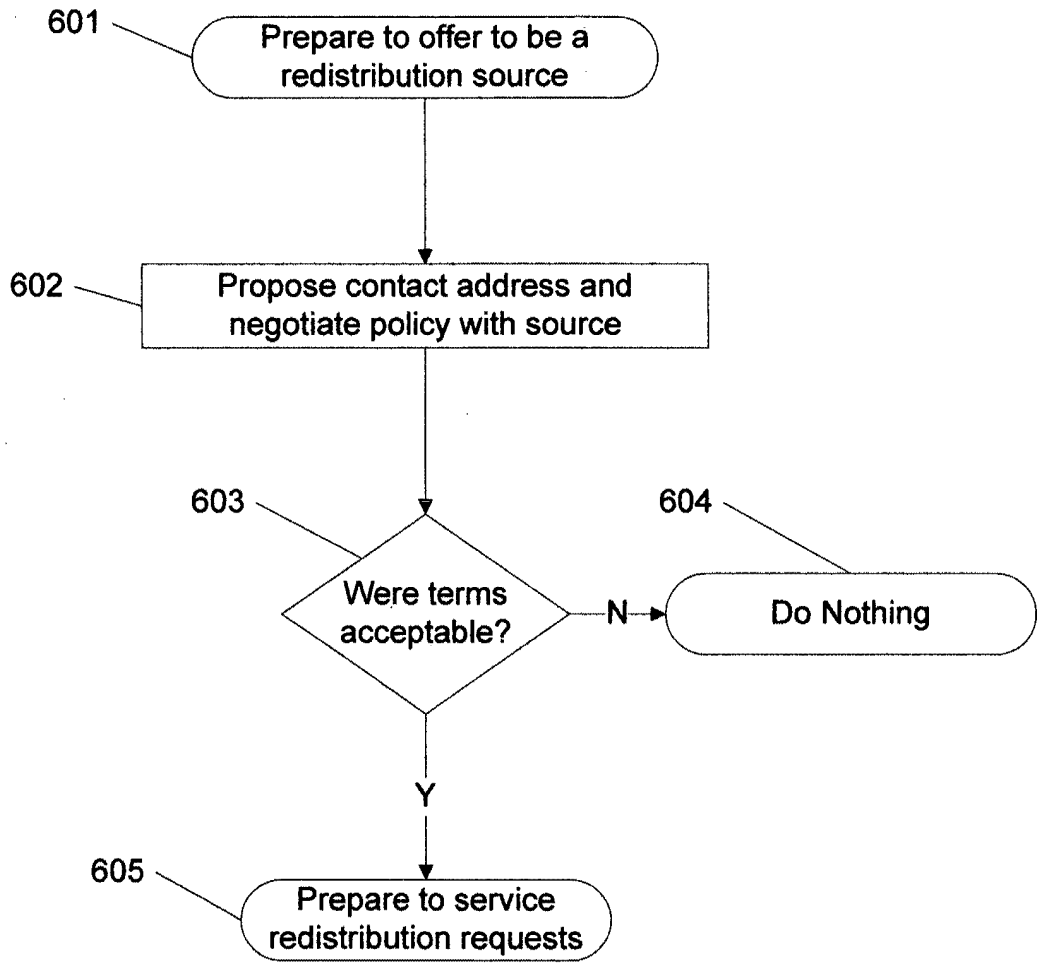
601
Prepare to offer to be a redistribution source
602
Propose contact address and negotiate policy with source
603
Were terms acceptable?
N
604
Do Nothing
Y
605
Prepare to service redistribution requests

Figure 6

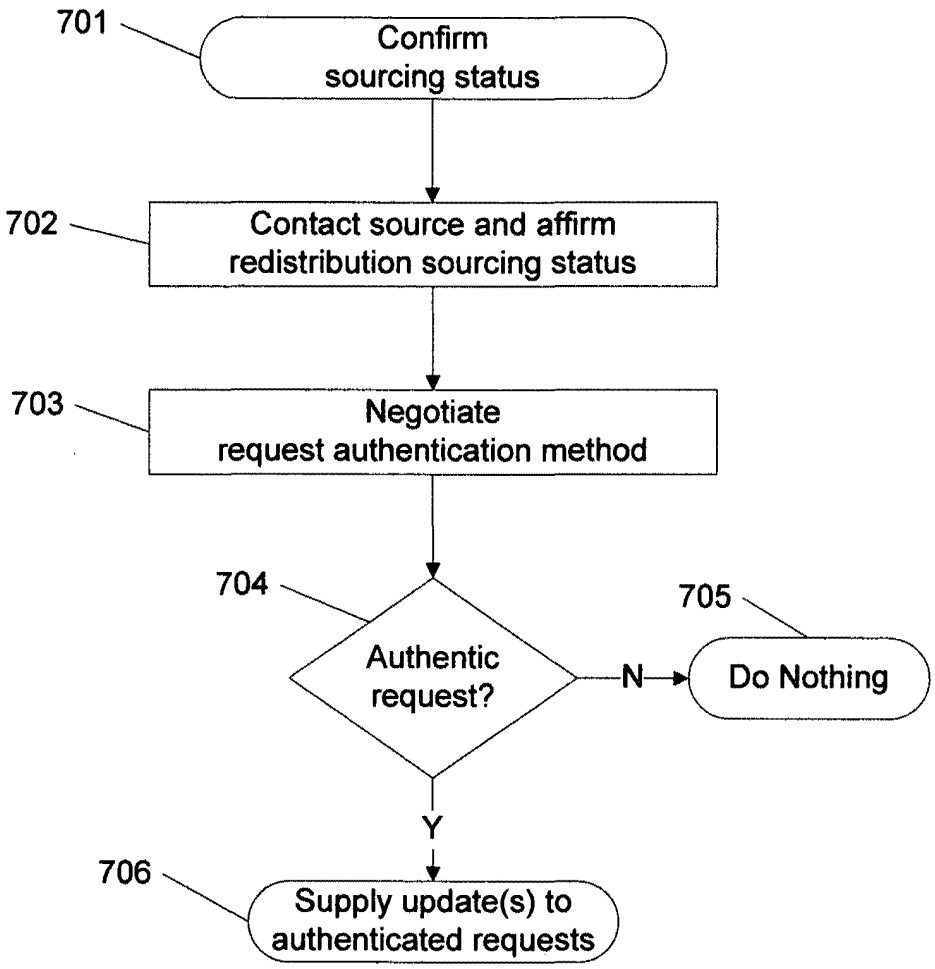
701
Confirm sourcing status
702
Contact source and affirm redistribution sourcing status
703
Negotiate request authentication method
704
Authentic request?
N
705
Do Nothing
Y
706
Supply update(s) to authenticated requests

Figure 7

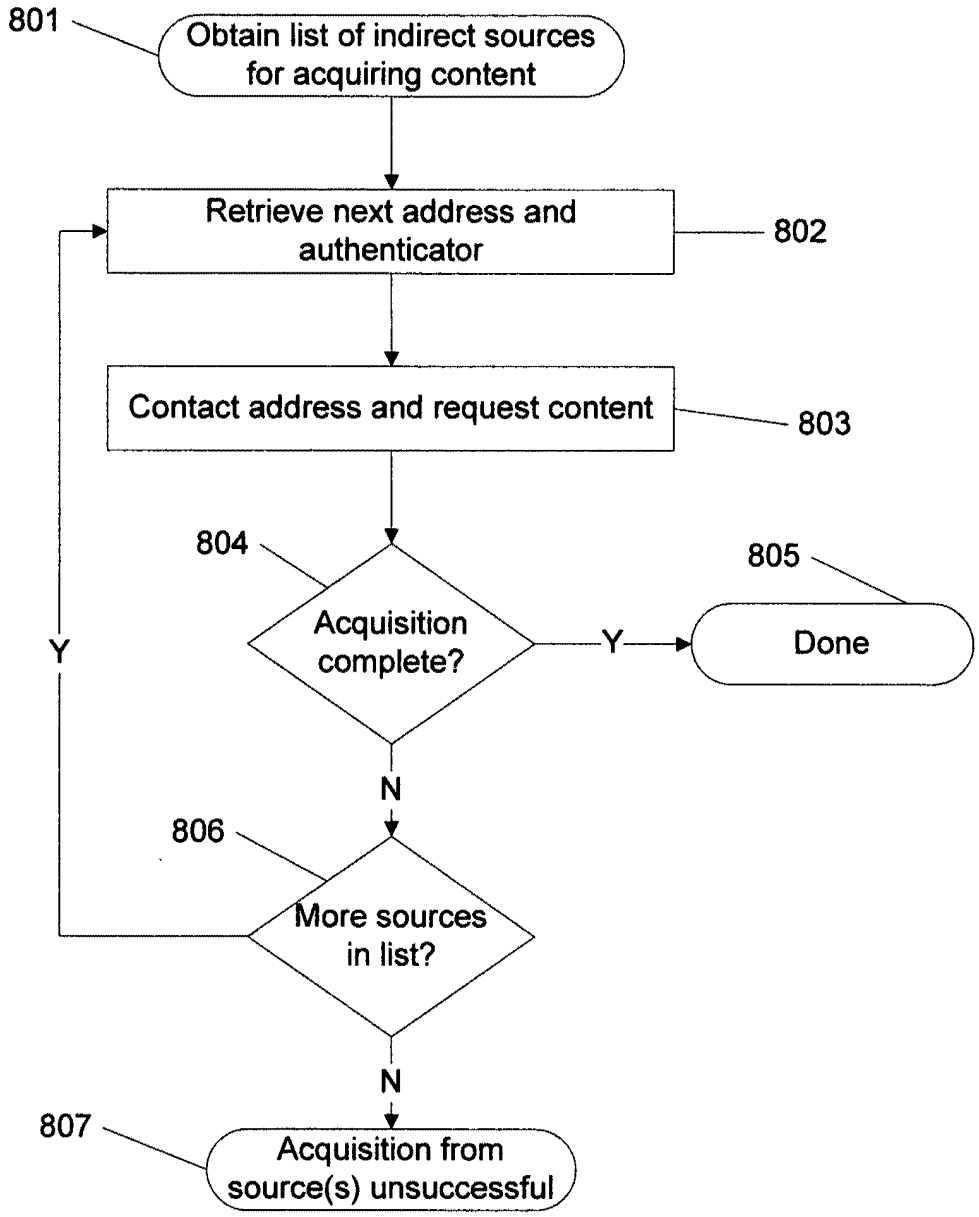
801
Obtain list of indirect sources for acquiring content
Retrieve next address and authenticator
802
Contact address and request content
803
804
Acquisition complete?
Y
805
Done
N
806
More sources in list?
Y
N
807
Acquisition from source(s) unsuccessful

Figure 8

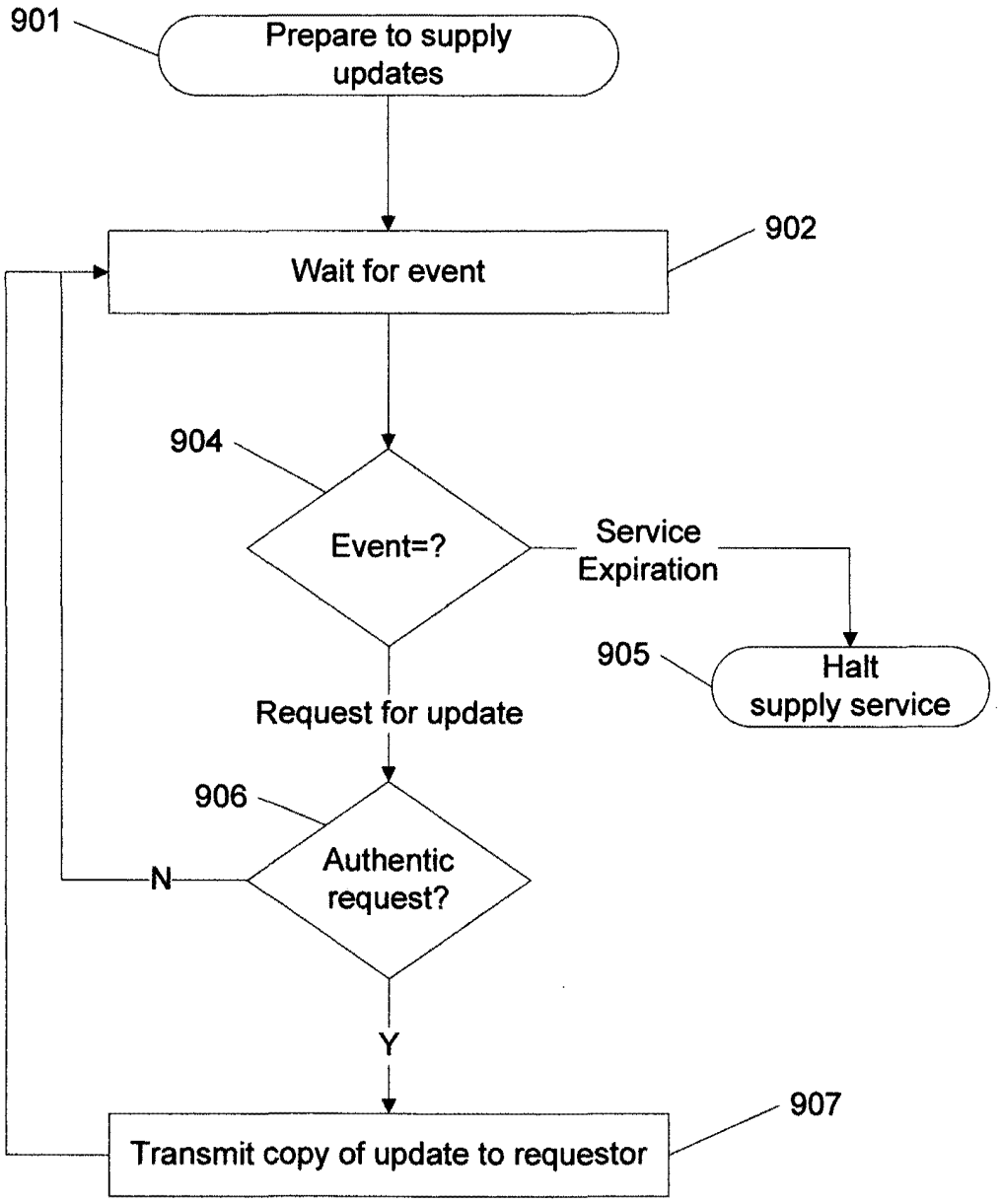
901
Prepare to supply updates
902
Wait for event
904
Event=?
Service Expiration
905
Halt supply service
Request for update
906
Authentic request?
N
Y
907
Transmit copy of update to requestor

Figure 9

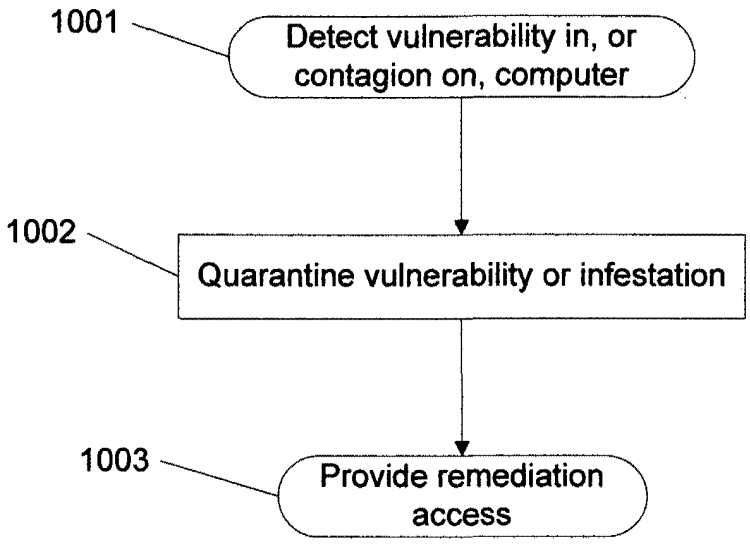
1001
Detect vulnerability in, or contagion on, computer
1002
Quarantine vulnerability or infestation
1003
Provide remediation access

Figure 10

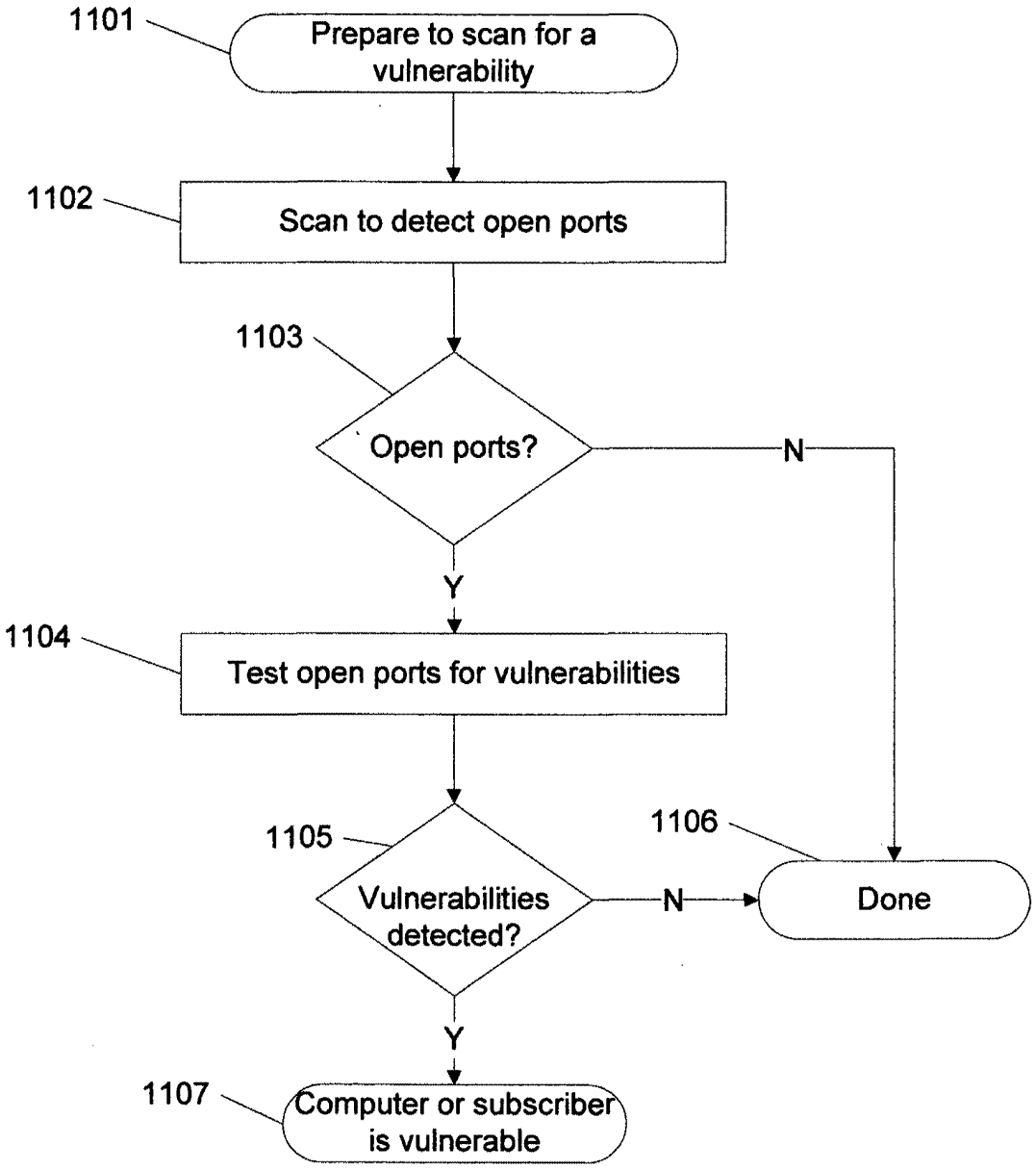
1101
Prepare to scan for a vulnerability
1102
Scan to detect open ports
1103
Open ports?
N
Y
1104
Test open ports for vulnerabilities
1105
Vulnerabilities detected?
N
1106
Done
Y
1107
Computer or subscriber is vulnerable

Figure 11

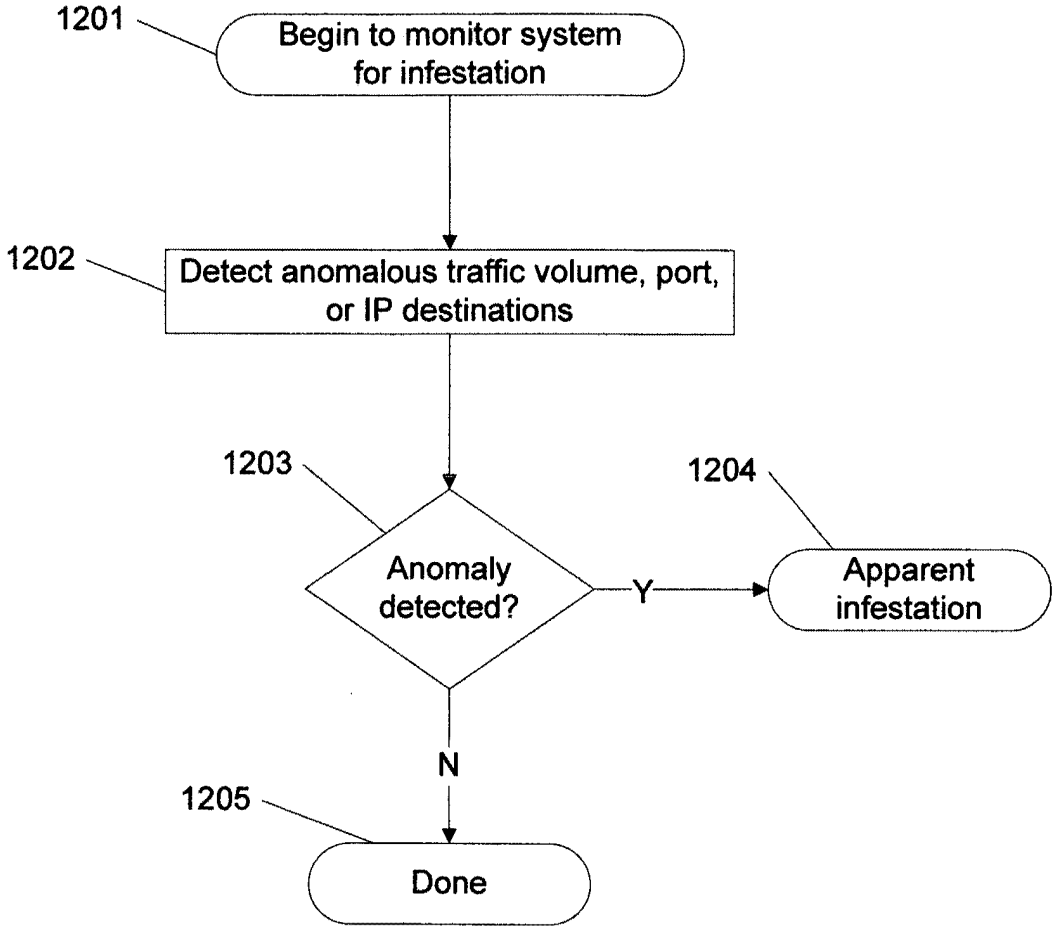
1201
Begin to monitor system for infestation
1202
Detect anomalous traffic volume, port, or IP destinations
1203
Anomaly detected?
Y
1204
Apparent infestation
N
1205
Done

Figure 12

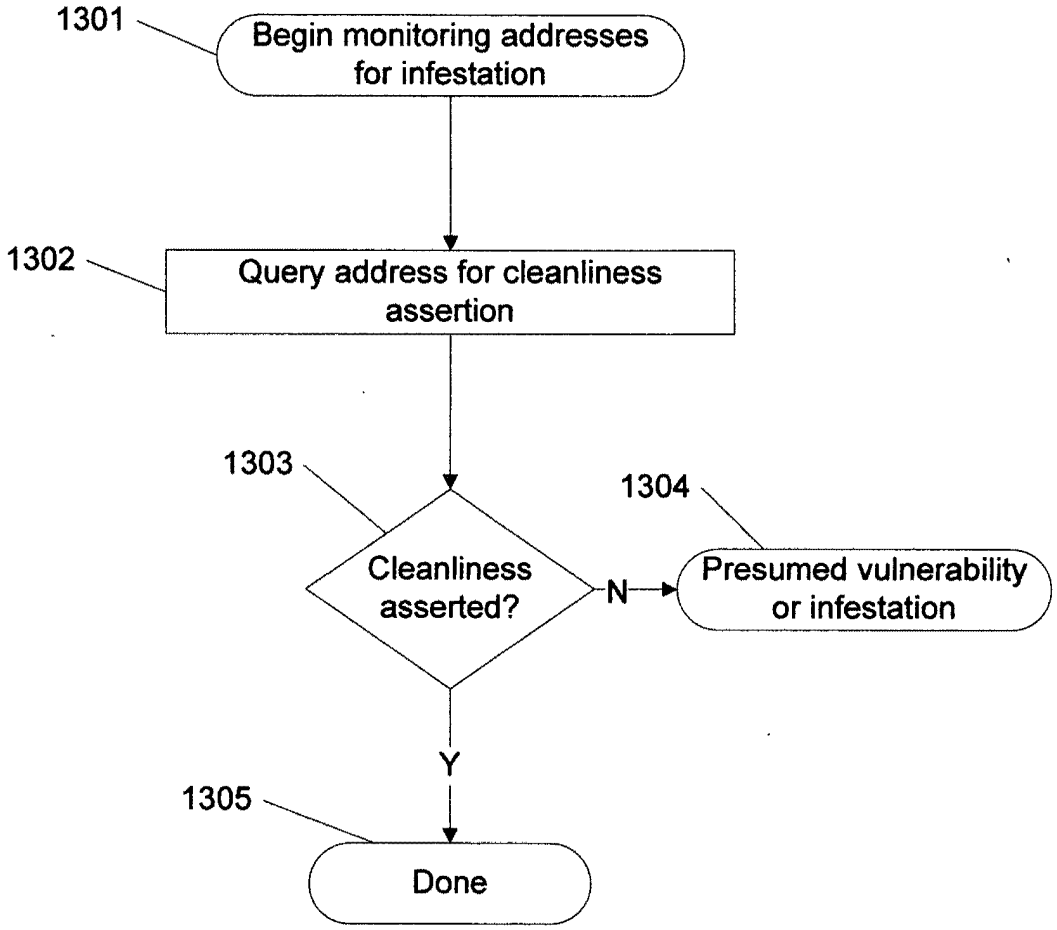
1301
Begin monitoring addresses for infestation
1302
Query address for cleanliness assertion
1303
Cleanliness asserted?
N
1304
Presumed vulnerability or infestation
Y
1305
Done

Figure 13

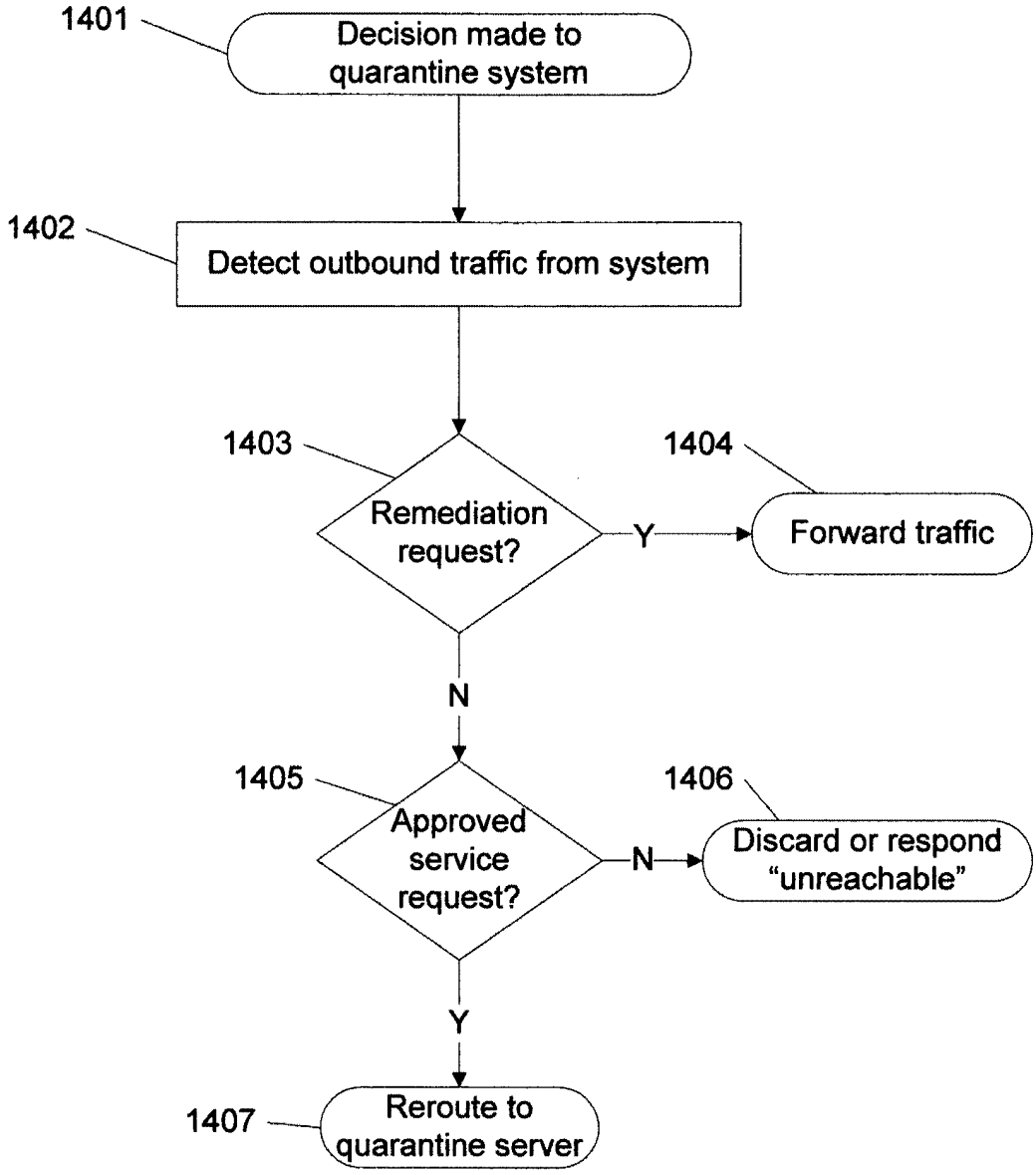
1401
Decision made to quarantine system
1402
Detect outbound traffic from system
1403
Remediation request?
Y
1404
Forward traffic
N
1405
Approved service request?
N
1406
Discard or respond "unreachable"
Y
1407
Reroute to quarantine server

Figure 14

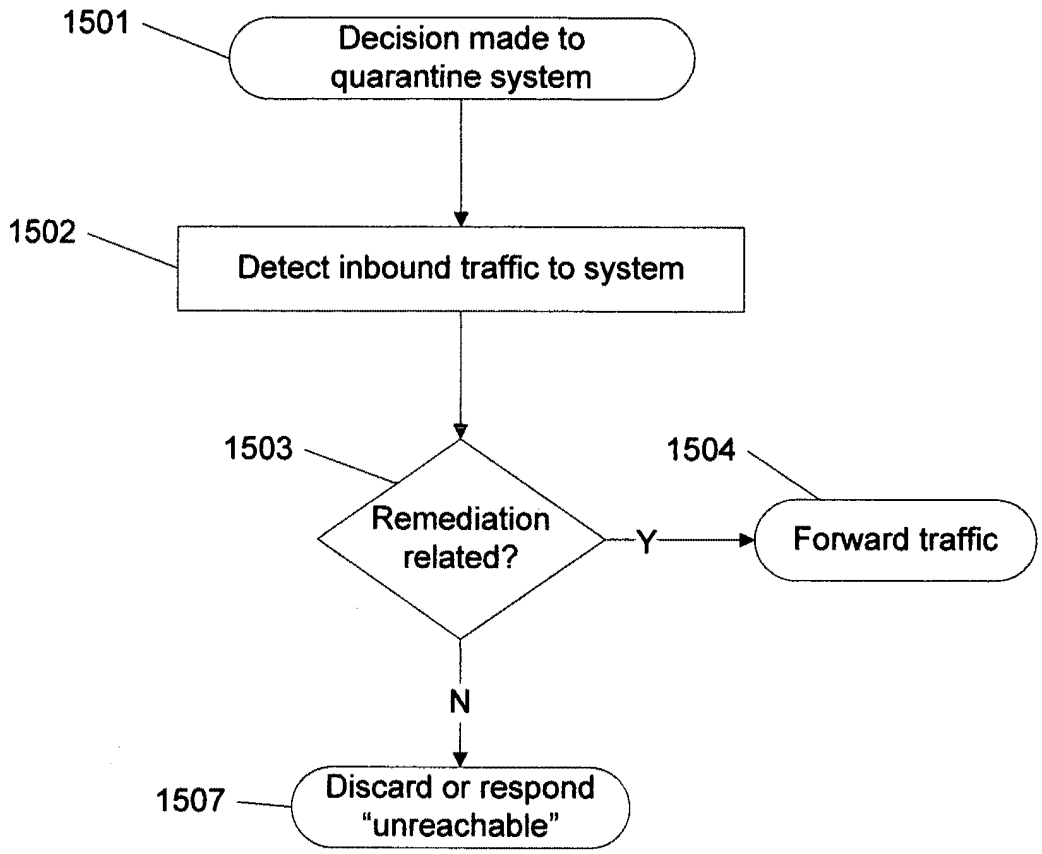
1501
Decision made to quarantine system
1502
Detect inbound traffic to system
1503
Remediation related?
Y
1504
Forward traffic
N
1507
Discard or respond "unreachable"

Figure 15

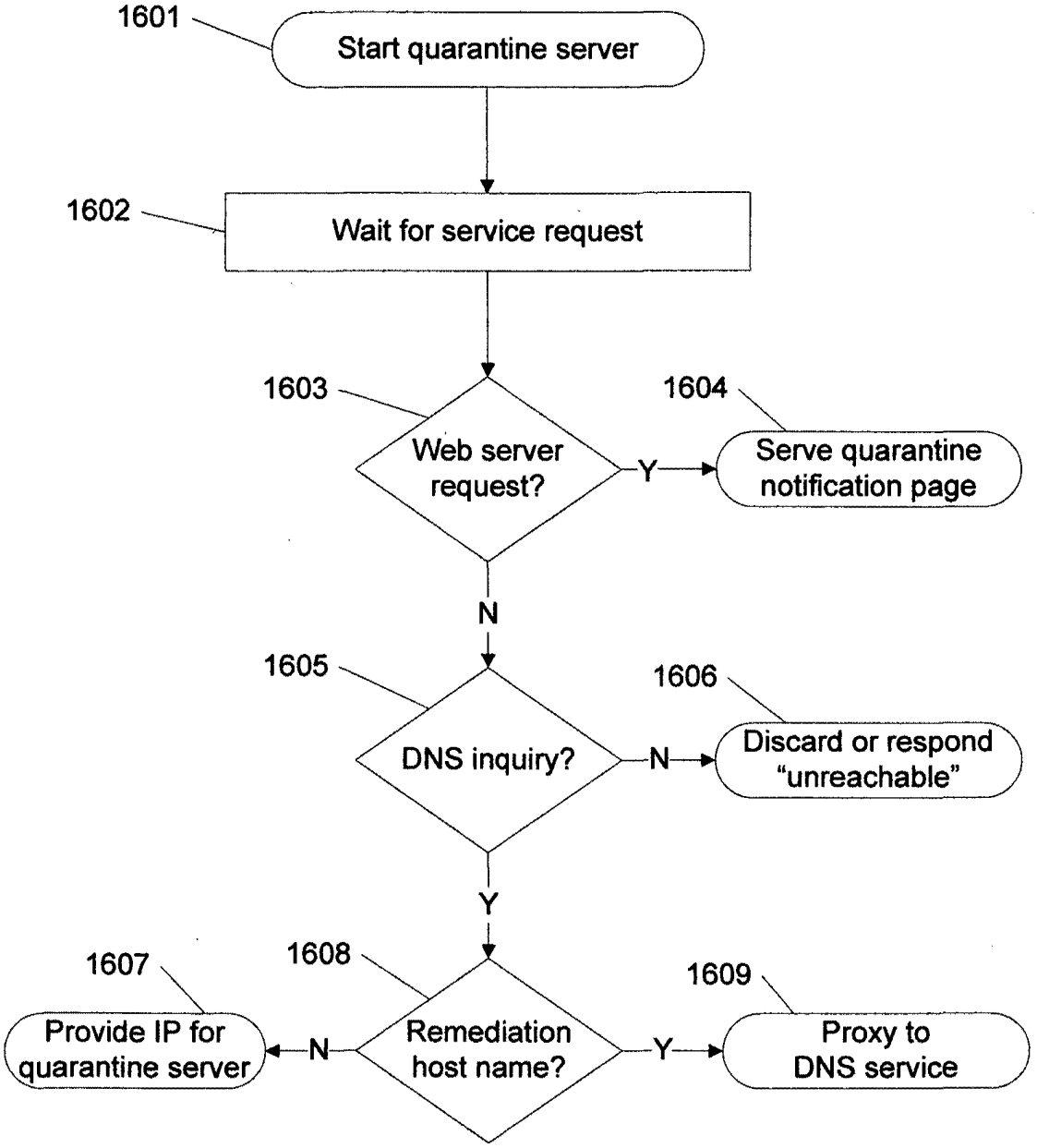
1601
Start quarantine server
1602
Wait for service request
1603
Web server request?
Y
1604
Serve quarantine notification page
N
1605
DNS inquiry?
N
1606
Discard or respond "unreachable"
Y
1608
Remediation host name?
N
1607
Provide IP for quarantine server
Y
1609
Proxy to DNS service

Figure 16

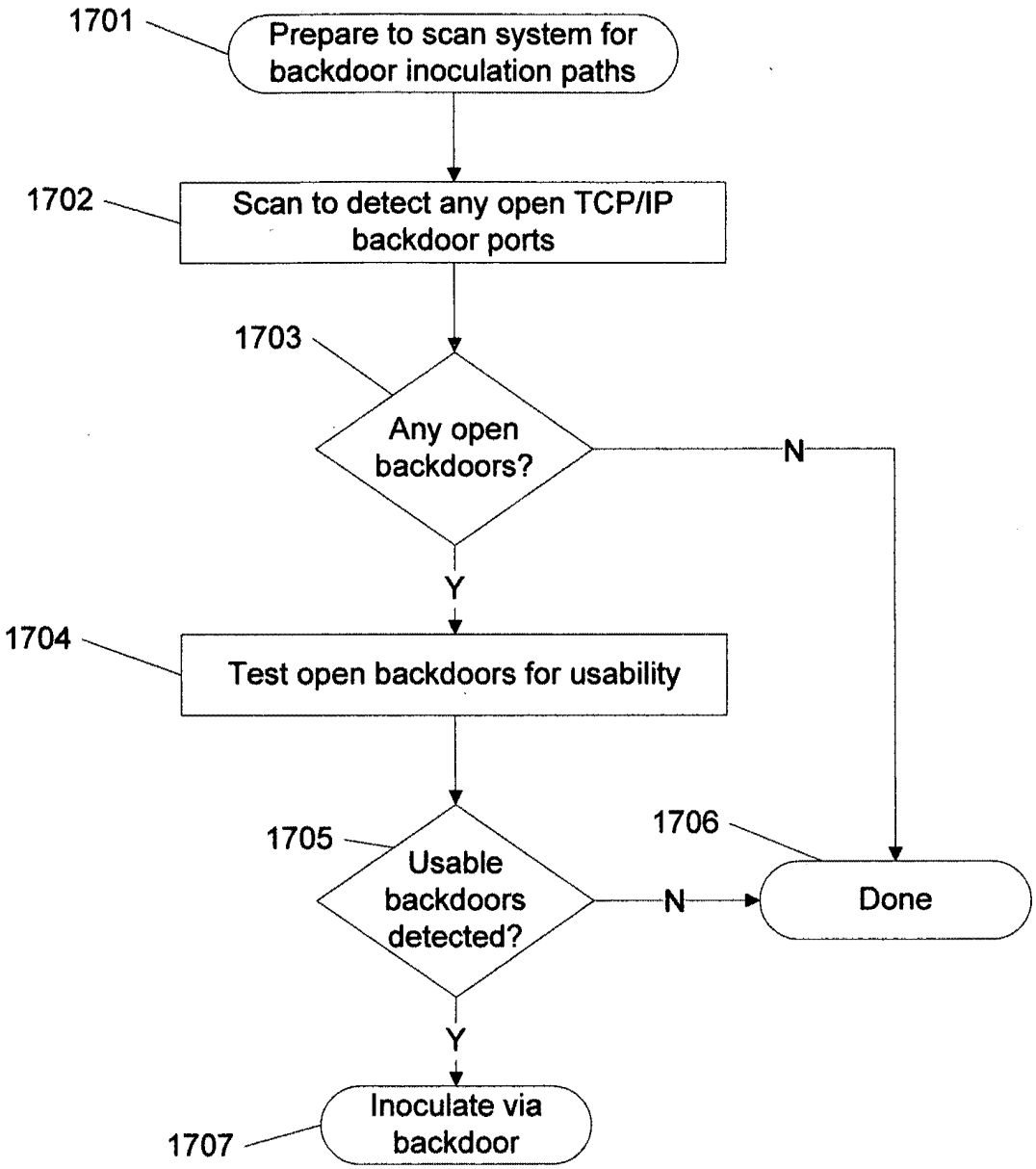
1701
Prepare to scan system for backdoor inoculation paths
1702
Scan to detect any open TCP/IP backdoor ports
1703
Any open backdoors?
N
Y
1704
Test open backdoors for usability
1705
Usable backdoors detected?
N
1706
Done
Y
1707
Inoculate via backdoor

Figure 17

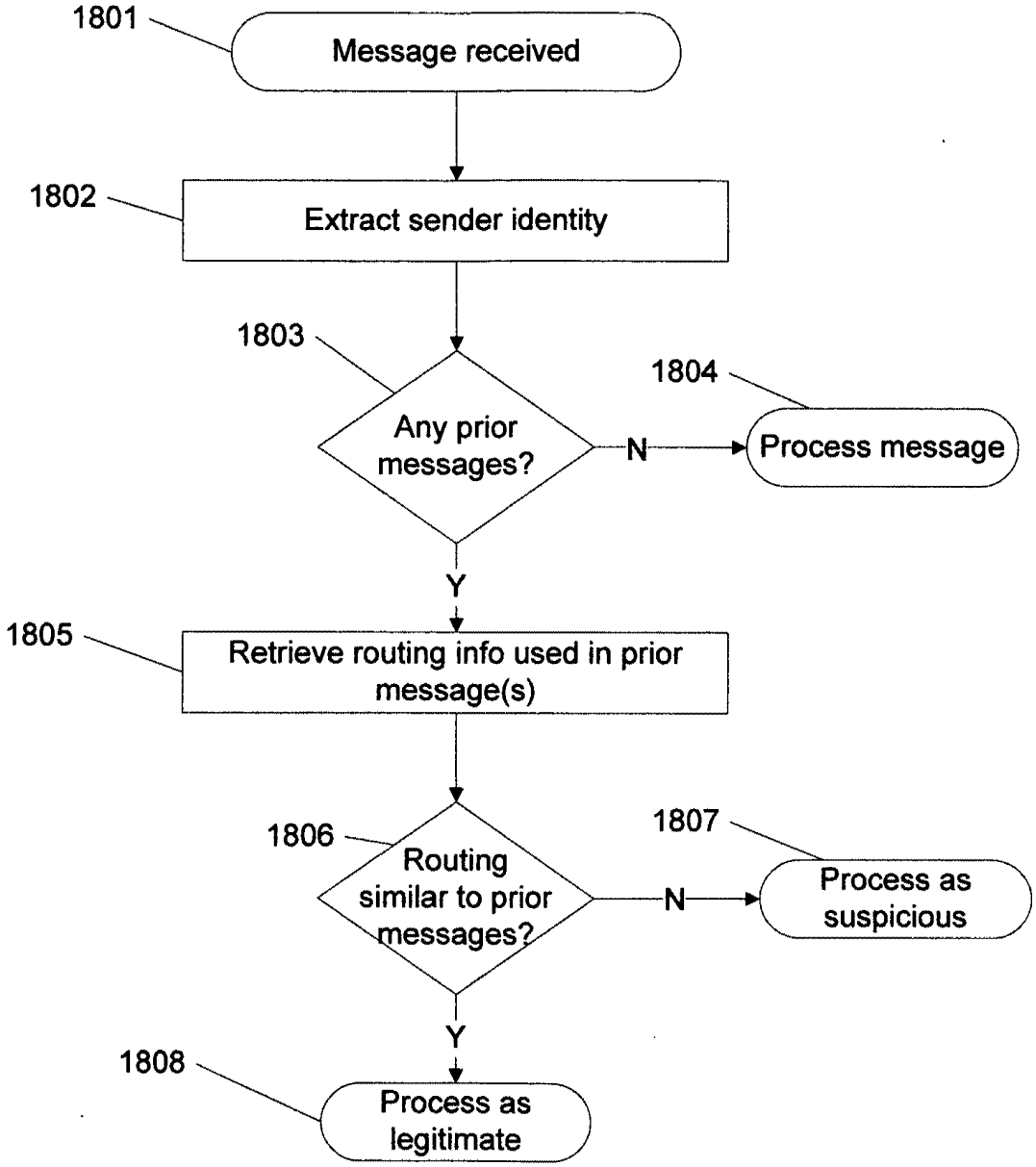
1801
Message received
1802
Extract sender identity
1803
Any prior messages?
N
1804
Process message
Y
1805
Retrieve routing info used in prior message(s)
1806
Routing similar to prior messages?
N
1807
Process as suspicious
Y
1808
Process as legitimate

Figure 18

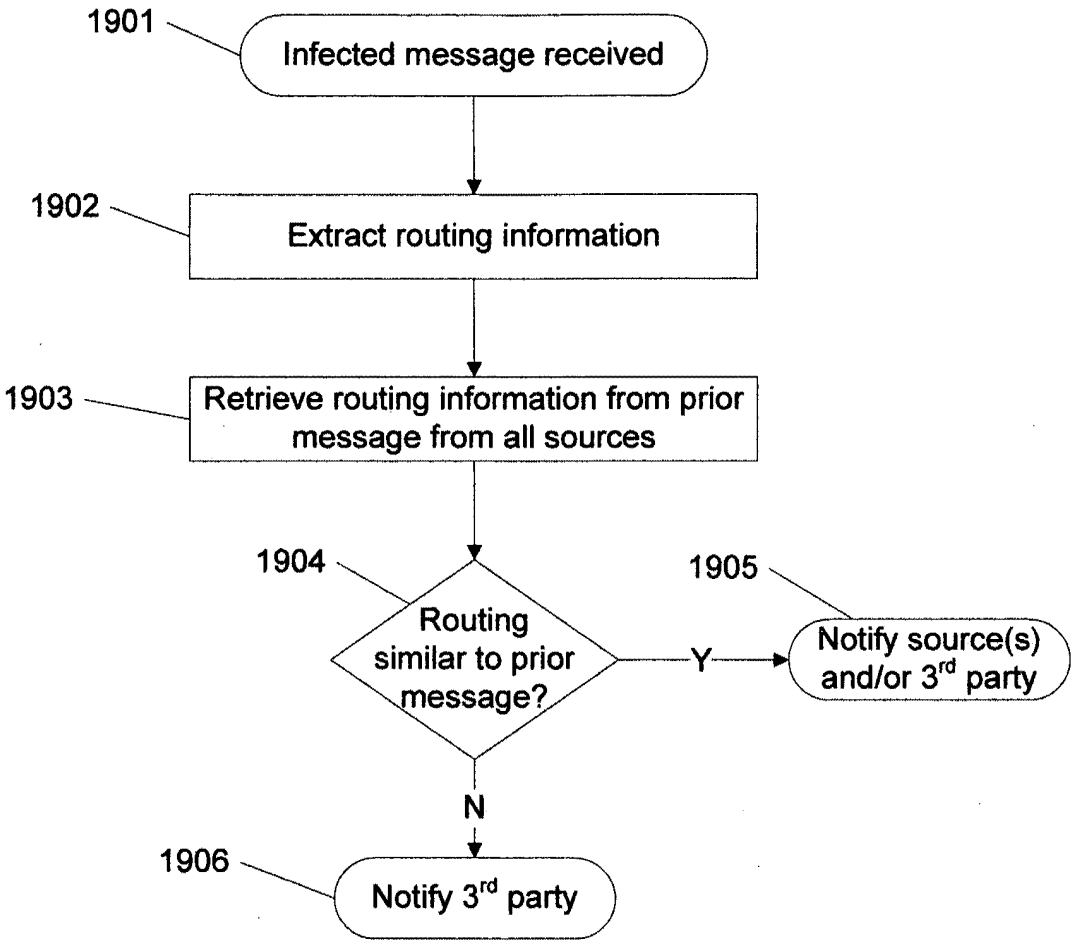
1901
Infected message received
1902
Extract routing information
1903
Retrieve routing information from prior message from all sources
1904
Routing similar to prior message?
Y
1905
Notify source(s) and/or 3rd party
N
1906
Notify 3rd party

Figure 19

PATENT APPLICATION SERIAL NO. _____

U.S. DEPARTMENT OF COMMERCE
PATENT AND TRADEMARK OFFICE
FEE RECORD SHEET

10/01/2004 EEKUBAY1 00000028 60613909

01 FC:2005 80.00 OP

PTO-1556
(5/87)

*U.S. Government Printing Office: 2002 — 489-267/69033