

**IN THE UNITED STATES DISTRICT COURT
FOR THE WESTERN DISTRICT OF TEXAS
AUSTIN DIVISION**

K.MIZRA LLC,

Plaintiff,

v.

GOOGLE LLC,

Defendant.

Civil Action No.: 1:25-

Jury Trial Demanded

COMPLAINT FOR PATENT INFRINGEMENT

Plaintiff K.Mizra LLC ("K.Mizra") files this Complaint for patent infringement against Defendant Google LLC ("Google"), alleging as follows:

I. INTRODUCTION

1. K.Mizra is a patent licensing company run by experienced management. The company focuses on high-value, high-quality patents and owns patent portfolios originating from a wide array of inventors, including patents and patent portfolios developed by well-known multinational corporations such as IBM, Intel, Rambus and others, as well as from research institutes such as Nederlandse Organisatie voor Toegespast Natuurwetenschappelijk Onderzoek (Netherlands Organization for Applied Scientific Research). By focusing on high-quality patents, K.Mizra provides a secondary market that enables inventors to recoup their research and development investments and to continue their innovations. K.Mizra offers licenses to its patents on reasonable terms and, in this way, plays an important role in the development of technologies that improve businesses and lives.

A. The Asserted Zero Trust Network Security Patents

2. K.Mizra is the owner by assignment of United States Patent Nos. 8,234,705 ("the '705 Patent") and 9,516,048 ("the '048 Patent") (collectively, "the Asserted Zero Trust Network Security Patents"). These Patents were involved in unsuccessful *Inter Partes* Review proceedings ("IPRs"), several now-resolved federal court litigations, and were originally invented by two highly-respected and prolific inventors—James A. Roskind and Aaron T. Emigh.

3. The Asserted Zero Trust Network Security Patents were originally owned by Dr. Roskind's and Mr. Emigh's company, Radix Labs, LLC. Dr. Roskind and Mr. Emigh were then, and remain today, focused on innovation, conducting new research, developing new technologies, and creating new and innovative computer products and systems.

4. Dr. Roskind, a co-inventor of these Patents, has bachelors, masters, and doctorate degrees from MIT in both electrical engineering and computer science and is a named inventor of over 300 U.S. patents. He has worked for Netscape as the Chief Architect and as the Netcenter Security Architect and was a co-founder of Infoseek, a company acquired by Disney for \$770 million. Dr. Roskind also was a key developer of Google's "transport protocol" that provides the tech giant billions of dollars in value every year.

5. Mr. Emigh, also a co-inventor of the Asserted Zero Trust Network Security Patents, graduated from the University of California, Santa Cruz with degrees in linguistics and computer and information sciences and is a named inventor of over 140 patents. Before working with Dr. Roskind, Mr. Emigh worked in various software-development positions, including software manager, architect, and engineer for Unicom and manager for the software development and technical marketing groups of Philips TriMedia. He has founded or co-founded many companies

in addition to Radix Labs, LLC, including CommerceFlow, Inc., which was acquired by eBay for the technology that Mr. Emigh helped develop.

6. After the Asserted Patents were issued, Dr. Roskind and Mr. Emigh recouped their research and development investment by selling certain technology rights and continued to work independently on their individual technological pursuits. K.Mizra ultimately acquired the Asserted Zero Trust Network Security Patents and licensed them to many major companies operating in the computer technology space. Some of those companies, including accused infringers, chose to test the validity of the Asserted Zero Trust Network Security Patents before settling their lawsuits involving those Patents. For example, accused infringers of the Asserted Zero Trust Network Security Patents previously sought IPRs by the Patent Trial and Appeal Board ("PTAB") of both of the Asserted Patents. A Final Written Decision ("Decision") issued in the IPR for the '705 Patent concluded that the petitioners had not shown by a preponderance of the evidence that the claims at issue were unpatentable. Based on the results of the '705 Patent IPR, an IPR for the similar '048 Patent was never instituted. The '705 Patent IPR Decision was appealed to the U.S. Court of Appeals for the Federal Circuit ("CAFC"), resulting in a procedurally-focused remand back to the PTAB. The PTAB recently dismissed the IPR Petition with prejudice.

7. The Asserted Patents were previously asserted by K.Mizra in this District against Cisco Systems, Inc. in *K.Mizra v. Cisco Systems, Inc.* (Civil Action No. 6:20-cv-01031-ADA), a case assigned to Judge Albright which resolved in September 2024, only a few weeks prior to trial being scheduled to commence. In that action, Judge Albright not only conducted a claim construction hearing and issued several important constructions of disputed claim terms, he also ruled on numerous summary judgment and *Daubert* motions as part of a final pretrial conference.

As such, Judge Albright is well-versed in the technology involved with the Asserted Zero Trust Network Security Patents, K.Mizra and other issues here likely to be involved.

8. K.Mizra remains ready, willing, and able to provide commercially-reasonable licenses for its various patented technologies to all entities who wish or need to use its covered technologies internally or in connection with products or services offered to others. As outlined below, Google is one such entity.

II. THE PARTIES

9. K.Mizra is a Delaware limited liability company with a mailing address of 777 Brickell Avenue, #500-96031, Miami, Florida 33131, and operates in Florida.

10. K.Mizra is the owner by assignment of the Asserted Patents.

11. Upon information and belief, Google is a corporation organized and existing under the laws of the state of Delaware, with a principal place of business at 1600 Amphitheatre Parkway, Mountain View, California 94043.

12. Upon information and belief, Google maintains an office in this District at 500 West Second Street, Austin, Texas 78701.

13. Upon information and belief, Google employs hundreds of people in its Austin, Texas office, including persons with knowledge of the Google products and technology at issue in this case.

14. Upon information and belief, Google is also registered to do business in Texas and may be served with process by serving Corporation Service Company, Google's registered agent in Texas located at 211 East Seventh Street, Suite 620, Austin, Texas 78701-3218.

III. JURISDICTION AND VENUE

15. This is an action for patent infringement under the patent laws of the United States, 35 U.S.C. §§ 1 *et seq.*, including 35 U.S.C. §§ 271, 281, and 284, among others. The Court has subject-matter jurisdiction over the claims raised in this action pursuant to 28 U.S.C. §§ 1331 and 1338(a).

16. This Court has personal jurisdiction over Google by virtue of, *inter alia*, its conduct of business in this District, its registration to do business in Texas, its appointment of a registered agent in Texas, its employment of hundreds of persons in its Austin office located in this District, and its substantial, continuous, and systematic contacts with the state of Texas and this District. Google intentionally markets and sells its infringing products directly and through agents to residents of Texas, enjoys substantial income from its business activities in the state of Texas and this District, and/or directly, by its own actions, and/or in combination with actions of customers and others under its control, has committed acts of patent infringement in this District at least by selling infringing products in this District.

IV. GENERAL ALLEGATIONS

A. The Asserted Zero Trust Network Security Patents

17. K.Mizra is the sole owner by assignment of the Asserted Zero Trust Network Security Patents with the full and exclusive right to bring suit to enforce them. K.Mizra is also entitled to sue to collect damages for all past infringement of the Asserted Zero Trust Network Security Patents.

18. The '705 Patent, titled "Contagion Isolation and Inoculation," was issued by the USPTO to Inventors Roskind and Emigh on July 31, 2012. A true and correct copy of the '705 Patent is attached hereto as Exhibit 1 and incorporated by reference.

19. The '048 Patent, titled "Contagion Isolation and Inoculation Via Quarantine," was issued by the USPTO to Inventors Emigh and Roskind on December 6, 2016. A true and correct copy of the '048 Patent is attached hereto as Exhibit 2 and incorporated by reference.

20. The Asserted Zero Trust Network Security Patents share similar (and in some respects, identical) specifications and claims, with both claiming priority to U.S. Provisional Application No. 60/613,909, filed on September 27, 2004 (the "Provisional Application").

B. Prior Licensing And Litigation Of The Asserted Patents

21. The Asserted Zero Trust Network Security Patents have been owned by several entities, in addition to Radix and K.Mizra, with some of those entities issuing to third parties certain rights to the technologies covered thereby.

22. K.Mizra has been involved in a number of actions it was required to institute to protect its patent rights, including actions involving the Asserted Zero Trust Network Security Patents. Most of those actions resulted in the execution of confidential patent license agreements.

23. Google is not and has never been a licensee of the Asserted Zero Trust Network Security Patents nor had or has any rights to use technologies covered by them. Google thus has no ownership or other rights (and is entitled to no rights) relating to the Asserted Zero Trust Network Security Patents.

C. Computer Network Security Problems In 2004 Solved By The Asserted Zero Trust Network Security Patents

24. The technology described in the Asserted Zero Trust Network Security Patents was invented by Dr. Roskind and Mr. Emigh, two colleagues living in the same area who had similar interests in innovating computer-related technologies. In 2003, the inventors decided to create a business—Radix Labs, LLC—which focused on developing intellectual property related to various computer technologies, including computer network security technologies. The inventors

focused on conceiving and reducing to practice inventions that they knew were needed (or soon would be needed) in the computer networking industry and then on drafting patent applications to capture and protect their technological innovations. In September of 2004, the inventors filed the Provisional Application to which both Asserted Zero Trust Network Security Patents claim priority. The Provisional Application described technology that focused on securing a computer network against the threats to which it was exposed when computer endpoints (e.g., laptop computers) were connected to a computer network. The Provisional Application, and by natural extension the Asserted Zero Trust Network Security Patents, also focus on remedying identified threats and quarantining those threats to mitigate any damage to the secured network.

25. Claims of the Asserted Zero Trust Network Security Patents are directed to technological solutions that address specific challenges grounded in computer network security. Maintaining the security of computer systems and networks is a tremendous concern for modern enterprises, since a breach of an internal network can have severe repercussions, including major financial losses, data theft, disclosure of sensitive information, network disruptions, data corruption, etc. The inventors of the Asserted Zero Trust Network Security Patents understood that while a network security appliance or hardware can be adept at keeping out unwanted external intrusions from the network, the most exploitable vulnerabilities of most networks are the end-user computers that roam throughout various public and private network domains, potentially exposing those computers to infection and then accessing and potentially infecting the entire and presumably secure computer network.

26. For example, the '705 Patent explains that

Laptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected network to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise,

or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in authorized ways and/or by unauthorized person.

See, e.g., Ex. 1 at 1:14–31.

27. The solution to these problems—as specified and claimed in the Asserted Zero Trust Network Security Patents—was an advanced departure from the conventional network access control solutions then in use and was then, as it remains today, patent eligible, highly valuable, novel, and non-obvious technology.

D. K.Mizra's Asserted Patent Claims Are Presumed Valid

28. K.Mizra asserts that at least, and without limitation, Claim 19 of the '705 Patent and Claim 17 of the '048 Patent have been directly infringed, either literally or under the doctrine of equivalents. K.Mizra reserves the right to assert additional claims of the Asserted Zero Trust Network Security Patents, including both independent and dependent claims, pursuant to the Court's (and other applicable) rules and procedures and as discovery progresses. These claims are referred to herein as the "Asserted Zero Trust Network Security Claims."

29. None of the Asserted Zero Trust Network Security Claims are directed to abstract ideas, and each employs inventive concepts and is directed to patent-eligible subject matter. All claims of the Asserted Zero Trust Network Security Patents are also presumed to be valid and enforceable against Google and others.

30. Indeed, the Asserted Zero Trust Network Security Patents' similar common specification and claims demonstrate that the need satisfied by the inventions of the Asserted

Claims was long-felt in the industry and thus unconventional. As one example, the '048 Patent provides that "[u]nwanted and/or malicious network communications, such as spam, phishing, work propagation, etc., hamper productivity and the use and enjoyment of computer and network resources by end users, burden affected networks with unauthorized and/or undesired traffic, and expose recipients to the risk of theft, fraud, etc." Ex. 2 at 1:22–27. The Asserted Zero Trust Network Security Patents' specification further provides that "[t]herefore, there is a need for an effective way to intercept and take corrective action with respect to unauthorized, unwanted, and/or otherwise malicious electronic mail and/or other network communications that better protects the network and provides protection to destination hosts that are not protected by destination or destination mail or messaging server-based filtering software." *Id.* at 1:46–52.

31. The specification (including the provisions quoted above), the figures (including those included below), and the text related to the figures further illustrate the complex, tiered network system architecture of the inventions captured by the Asserted Zero Trust Network Security Claims. These figures include the following:

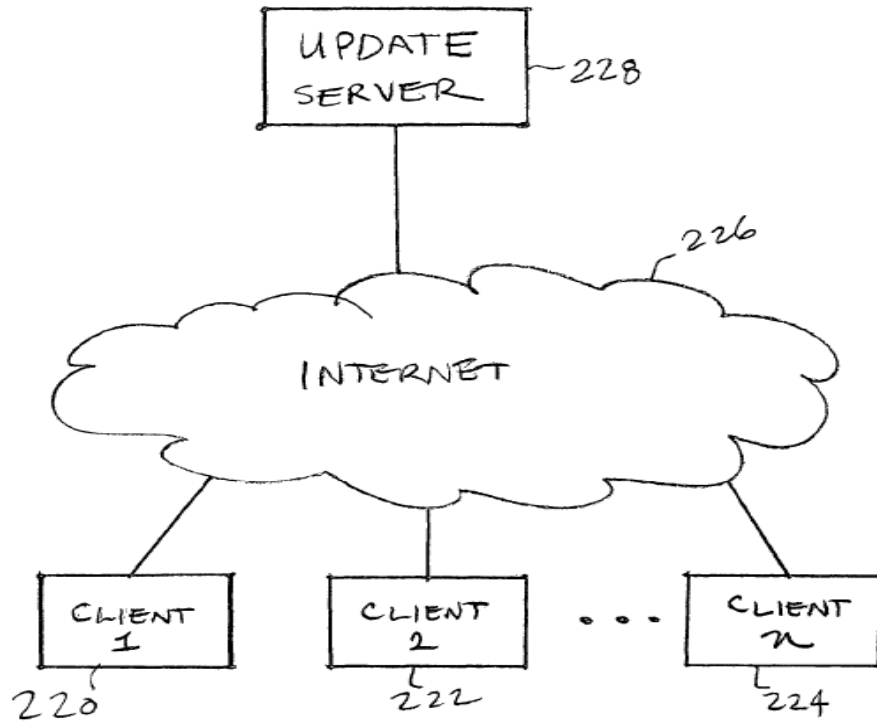
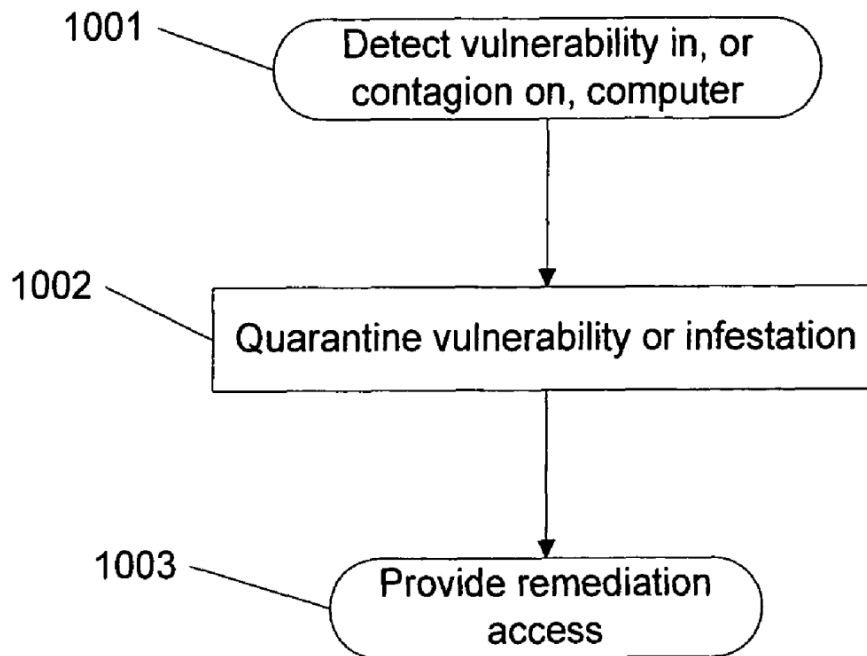


FIG. 2B

See Ex. 1 at Fig. 2B.



See *id.* at Fig. 10A.

32. The foregoing demonstrates that the inventions of the Asserted Zero Trust Network Security Claims focus on specific tamperproof hardware that must interact with unique software to improve network access control technology and protect a secure computer network and the data stored thereon from infected devices. As such, the Asserted Zero Trust Network Security Claims are eligible as a matter of law for patent protection under step one of *Alice Corp. v. CLS Bank Int'l*, 573 U.S. 208, 216 (2014).

33. All actions and steps recited in the Asserted Zero Trust Network Security Claims, including the act of quarantining endpoints or other computers, if necessary, require the involvement of various hardware components running dedicated software both before, during, and after the selection and isolation of an object. Said another way, a claim directed to allowing a machine to automatically and dynamically select and isolate an unsafe device attempting to access

a secure network is not simply adding a generic computer component to a fundamentally human process. Rather, it is removing the once-necessary human intervention from a fundamentally mechanical process, an "improvement in the functioning of a" networked system that simply cannot be considered directed to an abstract concept. *Enfish LLC v. Microsoft Corp.*, 822 F.3d 1327, 1339 (Fed. Cir. 2016).

34. As the specification confirms, the improvement captured by the Asserted Zero Trust Network Security Claims are not simply quarantining an infected device, but it is instead a multi-faceted network system involving multiple interrelated software and hardware components to protect a network from known and unknown threats. Specifically, the similar specifications of the Asserted Patents disclose that to reduce the burdens of having to manually identify, connect to, isolate, and remove malicious software from an infected device, the networked system can direct an unclean computer attempting to connect to the secure network, known as the host computer, to a form of remediation, such as downloading a software patch or a software update, removing material from the host computer and/or enabling certain settings, etc. present on the host computer. See Ex. 1 at 1:14–41. Indeed, the inventions of the Asserted Zero Trust Network Security Claims are each tethered to these advances over the art in the 2005 time frame, reciting methods and systems that automatically and dynamically detect an insecure condition by contacting a trusted computing base, receiving a response therefrom, determining if that response contains a valid identification of cleanliness, and configuring and implementing a remediation action based on what is discovered about the state of an endpoint or "host" computer. See, e.g., Ex. 1, Claims 12 and 19; Ex. 2, Claims 10 and 17. More specifically, the Asserted Zero Trust Network Security Claims require a system to communicate with a "trusted computing base" to determine when a response includes a valid digitally signed attestation of cleanliness, and to control access

to the network accordingly. These Asserted Zero Trust Network Security Claims are thus directed to a machine-implemented solution resolving a machine-specific problem, *i.e.* a machine's difficulty in detecting, isolating, and remediating infected endpoint devices (*e.g.*, host computers) to prevent contagion of and damage to the larger computer network.

35. The Asserted Zero Trust Network Security Claims are thus directed to a machine-implemented process for (1) determining whether the host computer is required to be quarantined, (2) isolating and inoculating the contagions (including directing the host to software programs and/or code designed to identify undesirable and/or unauthorized states) by quarantining the host, (3) limiting access to the network by the host computer so that the unsafe condition thereof can be remedied, and (4) allowing for remediation of an unsafe or infected host computer. As such, the Asserted Zero Trust Network Security Claims recite inventions with specific applications or improvements to technologies in the marketplace and cannot be considered abstract or patent ineligible under relevant law.

E. Failed IPRs

36. Fortune 100 companies accused of infringing the Asserted Zero Trust Network Security Patents have previously filed petitions for IPRs against each of the Asserted Patents, alleging that the claims of the Asserted Patents should be held invalid as either anticipated or obvious considering art not previously considered. Ultimately, the PTAB instituted an IPR against the '705 Patent, with similar third party IPRs that were subsequently filed being joined to the first-filed and instituted IPR.

37. The PTAB eventually issued its decision holding that no claims of the '705 Patent were unpatentable, finding that no asserted prior art reference alone or in combination satisfied the

limitation of "providing . . . an IP address of a quarantine server configured to serve the quarantine notification page" that was present in all claims of the '705 Patent.

38. Similarly, petitions for IPR were filed against the '048 Patent, but the PTAB denied institution of those, stating that the '048 Patent IPRs were "closely related" to the '705 Patent IPR petitions, that the petitions were based on and cited the same prior art and that "the challenged claims [were] materially the same" as those recited in the '705 patent's claims. The PTAB then offered that it had already "issued a Final Written Decision in [the '705 Patent's IPR], finding no claims of the '705 patent unpatentable" and that the '048 Patent IPR petitions were being denied institution for the same reasons.

39. The '705 Patent IPR Decision was then appealed to the CAFC, which reversed the PTAB's Decision on a few narrow procedural issues involving proof that the asserted prior art references would be combined by a person having ordinary skill in the art, as alleged by the petitioners.

40. A motion to dismiss the IPR was subsequently filed and the IPR has been dismissed with prejudice by the PTAB.

F. Google's Accused Instrumentalities And Services

41. Google has been making, selling, using, and offering for sale computer network access and security products, systems, and services that infringe the Asserted Zero Trust Network Security Patents in violation of 35 U.S.C. § 271 (collectively, "the Accused Zero Trust Network Security Instrumentalities"). These Accused Zero Trust Network Security Instrumentalities include, but are not limited to, Google Chrome Enterprise Premium, the sale, offer for sale, use, and/or manufacture in the United States of which constitutes infringement of the Asserted Zero Trust Network Security Claims, either literally or under doctrine of equivalents.

42. Upon information and belief, Google's Chrome Enterprise Premium product was based upon and formerly known as Google's "BeyondCorp Enterprise" product.

V. FIRST CLAIM FOR RELIEF
(Patent Infringement Under 35 U.S.C. § 271 of the '705 Patent)

43. K.Mizra incorporates paragraphs 1 through 42 as though fully set forth herein.

44. The '705 Patent includes 19 claims.

45. Google has infringed and is infringing one or more claims of the '705 Patent by making, importing, using, offering for sale, and/or selling the Accused Zero Trust Network Security Instrumentalities, in violation of 35 U.S.C. § 271(a).

46. Upon information and belief, and based upon publicly-available information, the Accused Zero Trust Network Security Instrumentalities meet each element of at least Claim 19 of the '705 Patent.

47. Claim 19 of the '705 Patent states:

[preamble] A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

[A] detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network,

[B] wherein detecting the insecure condition includes:

[B1] contacting a trusted computing base associated with a trusted platform module within the first host,

[B2] receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness,

[C] wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a

patch or a patch level associated with a software component on the first host;

[D] when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network,

[E] wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes

[E1] receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request,

[E2] and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

[F] permitting the first host to communicate with the remediation host.

Ex. 1 at 22:14–49.

48. Regarding the preamble of Claim 19, and to the extent the preamble is determined to be limiting (which it is not), the Accused Instrumentalities provide the features described in the preamble, which recites a "computer program product for protecting a network." For example, Google touts that its Chrome Enterprise product is "Google Cloud's zero-trust solution that enables an organization's workforce to access web applications securely from anywhere, without the need for VPN and without fear of malware, phishing, and data loss":

Chrome Enterprise Premium is Google Cloud's zero-trust solution that enables an organization's workforce to access web applications securely from anywhere, without the need for VPN and without fear of malware, phishing, and data loss.

By using Chrome Enterprise Premium, you can manage access for apps on Google Cloud, other clouds, and on-premises, define and enforce access policies based on user, device, and other contextual factors, and make apps more accessible and responsive through Google's global network.

See Ex. 3, Chrome Enterprise Premium documentation, p. 1 (available at <https://cloud.google.com/beyondcorp-enterprise/docs>) (last accessed mid Jan. 2025 and incorporated by reference). Additionally, Google's Chrome Enterprise products deliver endpoint (e.g., "host" computer) posture assessments and ensure that endpoints meet security and compliance policies before they are allowed to access a protected network. See *id.* Accordingly, and to the extent the preamble of Claim 19 is deemed limiting, the Accused Zero Trust Network Security Instrumentalities meet the limitation.

49. Limitation A of Claim 19 requires "detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network." The Accused Zero Trust Network Security Instrumentalities also meet all the requirements of limitation A of Claim 19. For example, Google's Chrome Enterprise products deliver endpoint posture assessments and ensure that endpoints meet security and compliance policies before they connect to the network:

Endpoint Verification overview



[Send feedback](#)

This document describes the basic concepts of Endpoint Verification.

Endpoint Verification lets security administrators or security operations professionals secure Google Cloud, on-premises apps and resources, and Google Workspace apps.

Endpoint Verification is part of Google Cloud [Chrome Enterprise Premium](#) and is available to all Google Cloud, Cloud Identity, Google Workspace for Business, and Google Workspace for Enterprise customers.

When to use Endpoint Verification

Use Endpoint Verification when you want an overview of the security posture of the devices that are used to access your organization's resources, such as laptops and desktops.

As a security administrator or security operations professional, your goal is to manage secure access to your organization's resources. The employees of your organization can use either the company-owned devices or their unmanaged personal devices to access the organization's resources. When Endpoint Verification is installed on the devices that access your organization's resources, it collects and reports device inventory information. You can use this device inventory information to manage secure access to your organization's resources.

When paired with the other offerings of Chrome Enterprise Premium, Endpoint Verification helps enforce fine-grained access control on your Google Cloud resources.

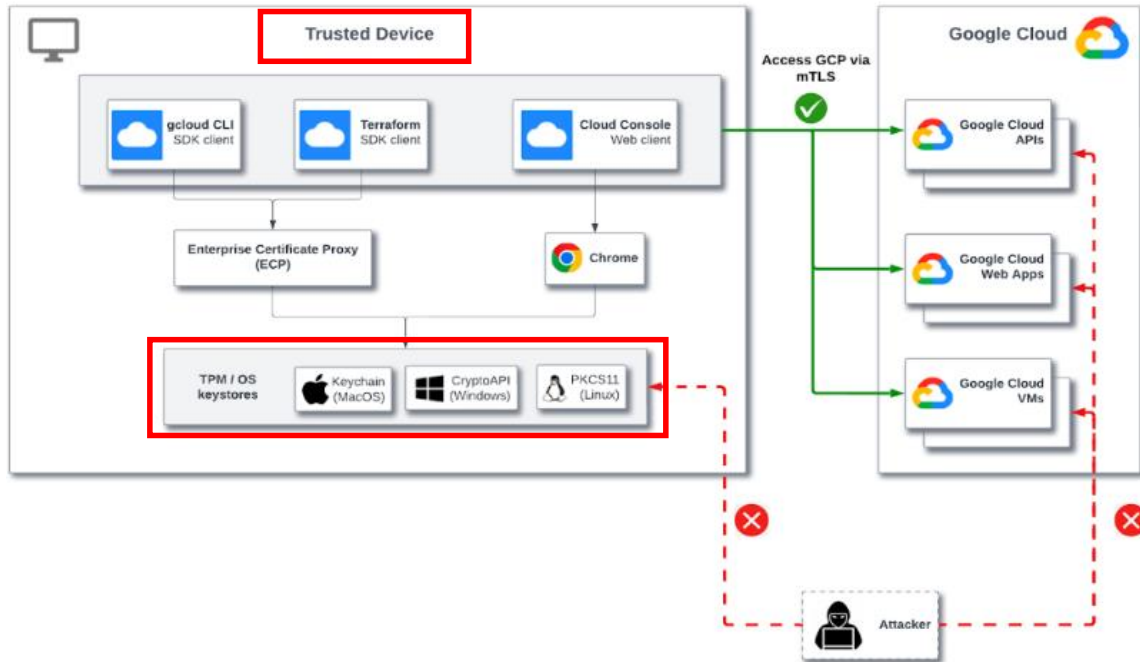
See Ex. 4, Endpoint Verification Overview, p. 1 (available at <https://cloud.google.com/endpoint-verification/docs/overview>) (last accessed mid Jan. 2025 and incorporated by reference).

Chrome Enterprise Premium presents a security model that allows for greater security posturing and policy for both applications and devices, while providing end users a better user experience no matter where they access from or what type of device they use to do so:

- For administrators:
 - Strengthen security posture to account for dynamic changes in a user's context.
 - Shrink the access perimeter to only those resources that an end user should be accessing.
 - Enforce device security postures for employees, contractors, partners, and customers for access, no matter who manages the devices.
 - Extend security standards with per-user session management and multifactor authentication.
- For end users:
 - Allow all end users to be productive everywhere without compromising security.
 - Allow the right level of access to work applications based on their context.
 - Unlock access to personally-owned devices based on granular access policies.

See Ex. 5, Chrome Enterprise Premium Overview, pp. 1–2 (available at <https://cloud.google.com/beyondcorp-enterprise/docs/overview>) (last accessed mid Jan. 2025 and incorporated by reference). Accordingly, the Accused Instrumentalities meet limitation A of Claim 19.

50. Limitation B1 of Claim 19 requires that "detecting [an] insecure condition includes . . . contacting a trusted computing base associated with a trusted platform module within the first host." The Accused Zero Trust Network Security Instrumentalities meet these requirements through, for example, the Google's Chrome Enterprise product which uses "strong key protection" such as "secure cryptographic storage such as TPMs and OS keystores." See Ex. 6, How To Prevent Account Takeovers With New Certificate-Based Access, p. 1 (available at <https://cloud.google.com/blog/products/identity-security/how-to-prevent-account-takeovers-with-new-certificate-based-access>) (last accessed mid Jan. 2025 and incorporated by reference).



See *id.* at 2. The Accused Zero Trust Network Security Instrumentalities obtain information from the host computer through digitally-signed certificates to determine whether the host computer is secure and can be trusted to access the protected network:

Certificate-based access can help you build a multi-layered defense against account takeovers, bolster data security, and maintain user trust. At Google, we have used this unique Zero Trust approach for many years as a strong defense to protect our technical infrastructure and employees. Now with CBA, we are extending this same level of security to our Google Cloud customers. Here are the key attributes of this approach:

- **Certificate-based access control:** Granular access policies with X.509 device certificates ensures only legitimate users with the correct certificate can access cloud resources.
- **Protections beyond initial login:** In contrast to using mTLS only for authentication, CBA also evaluates every authorization request to help safeguard resource access.
- **Strong key protection:** CBA's use of mTLS leverages secure cryptographic storage such as TPMs and OS keystores. Tooling such as Enterprise Certificate Proxy (ECP) are offered to users that empower them to safeguard private keys helping to ensure keys remain inaccessible to attackers without physical device access.

CBA allows you to enforce certificate-based authentication for all of Google Cloud, including specific resources (VM, Console, API), individual groups or organizations, across multiple end user vectors from browser to gcloud and terraform command-line-interfaces. It is integrated with many services in Google Cloud for effective access enforcement to cloud resources. For example, Context-Aware Access for Google APIs and the Cloud Console, Identity Aware Proxy (IAP) for web apps and workloads, and VPC Service Controls (VPC-SC) for network enforcement.

See *id.* at 4. Therefore, the Accused Zero Trust Network Security Instrumentalities meet limitation B1 of Claim 19.

51. Limitation B2 requires that "detecting the insecure condition" also includes "receiving a response and determining whether the response includes a valid digitally signed attestation of cleanliness." The Accused Zero Trust Network Security Instrumentalities also meet all the requirements of limitation B2, as the Chrome Enterprise products receive from the host computer requested information and then determines, based on the information received, whether a user (*i.e.*, a first "host") attempting to access cloud resources (*i.e.*, a protected network) is secure and trusted. See, *e.g.*, *id.* This process requires back-and-forth communication between a host computer and the Chrome Enterprise products about the cleanliness of the host computer/endpoint device:

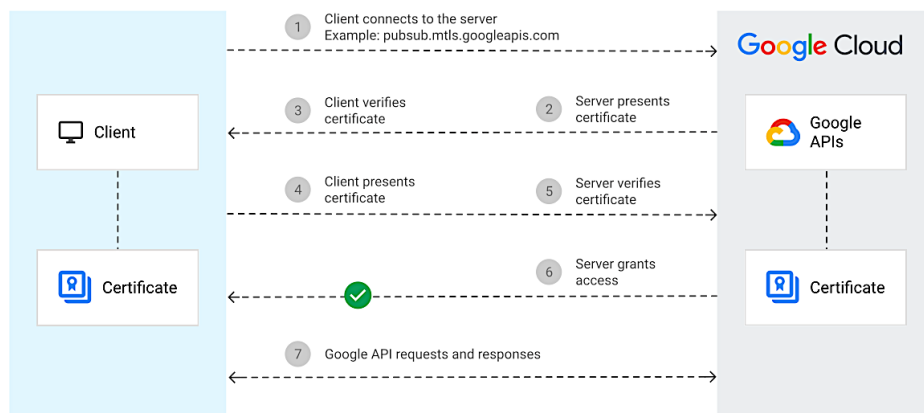
How the Google APIs validate device identity

The TLS protocol uses a technique called public key infrastructure (PKI), which relies on a pair of asymmetric keys: a public key and a private key. Anything encrypted with the private key can be decrypted only with the public key. The Google Cloud APIs use the TLS protocol to verify the identity of a device by decrypting the message encrypted by the private key using the public key of the certificate during the mTLS handshake. The successful decryption proves the possession of the private key which is only available from trusted devices.

To enable the mTLS handshake and validation process, a client must do the following:

- Establish an mTLS connection with the Google APIs by using mTLS-specific API endpoints. The mTLS-specific endpoints have the following format: `[service].mtls.googleapis.com`
- Discover and use the device certificate during the mTLS handshake. If you are using Endpoint Verification for certificate deployment, this type of certificate is automatically discovered and used by the supported clients.

The following diagram illustrates the mTLS handshake between a client and a Google API server:



See Ex. 7, Understand Mutual TLS at Google Cloud, p. 1 (available at <https://cloud.google.com/beyondcorp-enterprise/docs/understand-mtls>) (last accessed mid Jan. 2025 and incorporated by reference). Additionally, Google's Chrome Enterprise Premium products check digital signatures to determine the cleanliness of the host computer:

Deploy Endpoint Verification to use with certificate-based access

[Send feedback](#)

As part of a Chrome Enterprise Premium solution, Endpoint Verification provides critical device trust and security-based access control, and can help enforce fine-grained access control on your Google Cloud resources.

See Ex. 8, Deploy Endpoint Verification To Use With Certificate-Based Access, p. 1 (available at <https://cloud.google.com/beyondcorp-enterprise/docs/deploy-cba-endpoint-verification>) (last accessed mid Jan. 2025 and incorporated by reference). Thus, the Accused Zero Trust Network Security Instrumentalities meet limitation B2 of Claim 19.

52. Limitation C requires that "the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host." The Accused Zero Trust Network Security Instrumentalities meet these requirements as the Chrome Enterprise products check the compliance of each endpoint device attempting to connect to the protected network by performing an endpoint control check that involves matching the endpoint configuration parameters received from the endpoint device with specific device profile attributes, such as antimalware programs and applications:

Google announced a new feature and administration serverside API for the Chrome OS called Verified Access. This API will cryptographically validate the identity of any Chrome OS device that has connected to an enterprise's network. Additionally, it will allow the enterprise to verify that any connected devices conform to the company's security policies.

The new Google API uses digital certificates stored in the hardware-based Trusted Platform Modules (TPMs) that are found in every Chrome OS device. The TPM creates a way to ensure the correct security state of the devices has not been altered.

See Ex. 9, New Google API Will Securely Verify Chrome Devices, p. 2 (available at <https://securityintelligence.com/news/new-google-api-will-securely-verify-chrome-devices/>) (last

accessed mid Jan. 2025 and incorporated by reference). Accordingly, the Accused Zero Trust Network Security Instrumentalities meet limitation C of Claim 19.

53. Limitation D requires that "when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network." The Accused Zero Trust Network Security Instrumentalities further meet these requirements by quarantining noncompliant, *i.e.*, unclean, endpoint devices attempting to connect to the protected network:

Using Context-Aware Access, admins can set up different access levels based on a user's identity and the context of the request (location, device security status, IP address). This can help provide granular access controls without the need for a VPN, and give users access to Google Workspace resources based on organizational policies. Insights and recommendations help admins improve the cybersecurity posture of their organization by proactively identifying areas that need attention, significantly reducing the need for admins to identify these risks themselves. For example, if we detect devices with outdated operating system versions accessing corporate Workspace data, we can surface this as an Insight & pair it with a recommendation to block such devices from accessing Workspace data with a few clicks.

See Ex. 10, Context Aware Access Insights and Recommendations Are Now Generally Available, p. 3 (available at <https://workspaceupdates.googleblog.com/2024/10/context-aware-access-insights-and-recommendations.html>) (last accessed mid Jan. 2025 and incorporated by reference).

Using context-aware access, you now have the option to automatically block access to Google Workspace data from compromised Android and iOS devices. A device may be counted as compromised if certain unusual events are detected, including devices that are jailbroken, bypassing of security controls, modification of restricted settings, and more.

See Ex. 11, Block Compromised Mobile Devices Using Context-Aware Access, p. 2 (available at <https://workspaceupdates.googleblog.com/2024/05/block-compromised-mobile-devices-using-context-aware--access.html>) (last accessed mid Jan. 2025 and incorporated by reference).

Accordingly, the Accused Zero Trust Network Security Instrumentalities meet limitation D of Claim 19.

54. Limitation E1 requires that "preventing the first host from sending data to one or more other hosts associated with the protected network includes . . . receiving a service request sent by the first host [and] serving a quarantine notification page to the first host when the service request comprises a web server request." The Accused Zero Trust Network Security Instrumentalities meet these requirements because the Chrome Enterprise products are configured to determine if an endpoint device matches the profile designated for such a device and if the endpoint device does not match, it is quarantined and restricted from accessing the protected network. The quarantine prevents the user (*i.e.* first host) from sending data to the protected network by blocking the user until the user fixes the problem causing the user to be blocked:

Using remediation messages and custom messages in Context-Aware Access, you can help users unblock themselves when a policy prevents them from accessing an app. These optional (but recommended) messages can help get users back to productivity and reduce support calls for admins.

For example, say that a user on a mobile device is using Gmail in the office successfully during the day, but is blocked when they try to access Gmail at home in the evening. When remediation messages are enabled, they will see guidance on how to address the reason they are blocked.

Remediation and custom messages are supported for access levels created in both Basic mode and Advanced mode. Also, they are supported for both Core Services and SAML apps.

See Ex. 12, Allow Users to Unblock Apps With Remediation Messages in Context Aware Access, p. 1 (available at <https://support.google.com/a/answer/11560430?sjid=18176077754207218303-EU>) (last accessed mid Jan. 2025 and incorporated by reference). A quarantine message is also delivered to the unclean endpoint device notifying its user of the quarantine.

Use remediation messages and custom messages to help your users unblock themselves

When blocked, your users can encounter:

- **Default message**—Displays if you have not added remediation messages or custom messages. An example default message is: **Your organization's policy is blocking access to this app.**
- **Remediation messages**—Replaces the default message. The messages are system generated, and correspond to the specific policy violation that blocked the user. Remediation messages can present several remediation options to the user, which they can expand by clicking **Show more options**. In the case of several remediation options, the user needs to complete the steps for any one of the available options to unblock themselves.
- **Custom message**—Adds specific help for the user, such as additional advice on getting unblocked or a helpful link to click. You add custom messages as needed. A custom message can appear in conjunction with the default message, or with remediation messages.

See *id.* Accordingly, the Accused Zero Trust Network Security Instrumentalities meet limitation E1 of Claim 19.

55. Limitation E2 requires that "preventing the first host from sending data to one or more other hosts associated with the protected network includes" "in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition." The Accused Zero Trust Network Security Instrumentalities also meet all the requirements of limitation E2 of Claim 19. For example, Chrome Enterprise provides the user with a quarantine notification page containing links, or IP address(es), with resources (*i.e.* quarantine servers) configured to resolve the quarantine. In other words, Chrome Enterprise provides remediation information to bring the device into compliance so that it can access the protected network.

Use remediation messages and custom messages to help your users unblock themselves

When blocked, your users can encounter:

- **Default message**—Displays if you have not added remediation messages or custom messages. An example default message is: **Your organization's policy is blocking access to this app.**
- **Remediation messages**—Replaces the default message. The messages are system generated, and correspond to the specific policy violation that blocked the user. Remediation messages can present several remediation options to the user, which they can expand by clicking **Show more options**. In the case of several remediation options, the user needs to complete the steps for any one of the available options to unblock themselves.
- **Custom message**—Adds specific help for the user, such as additional advice on getting unblocked or a helpful link to click. You add custom messages as needed. A custom message can appear in conjunction with the default message, or with remediation messages.

See *id.* Accordingly, the Accused Zero Trust Network Security Instrumentalities meet limitation E2 of Claim 19.

56. Limitation F requires "permitting the first host to communicate with the remediation host." The Accused Zero Trust Network Security Instrumentalities meet these requirements as the Chrome Enterprise products allow a quarantined endpoint device to access remediation resources to help make the device complaint. *See id.* Accordingly, the Accused Zero Trust Network Security Instrumentalities meet limitation F of Claim 19.

57. Google's acts of infringement have occurred within this District and elsewhere throughout the United States.

58. As a result of Google's infringing conduct, K.Mizra has suffered damages. Google is liable to K.Mizra in an amount that adequately compensates K.Mizra for Google's infringement in an amount that is no less than a to-be-calculated fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

VI. SECOND CLAIM FOR RELIEF
(Patent Infringement Under 35 U.S.C. § 271 of the '048 Patent)

59. K.Mizra incorporates paragraphs 1 through 58 as though fully set forth herein.

60. The '048 Patent includes 20 claims.

61. Google has directly infringed one or more claims of the '048 Patent by making, importing, using, offering for sale, and/or selling the Accused Zero Trust Network Security Instrumentalities, in violation of 35 U.S.C. § 271(a).

62. Based on publicly available information, the Accused Zero Trust Network Security Instrumentalities satisfy every limitation of at least Claim 17 of the '048 Patent.

63. Claim 17 of the '048 Patent recites the following:

[preamble] A computer program product, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

[A] detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network,

[B] wherein detecting the insecure condition includes

[B1] contacting a trusted computing base associated with a trusted platform module within the first host,

[B2] receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness,

[C] wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

[D] when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network,

[E] wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes

[E1] receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request,

[E2] wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

[F] permitting the first host to communicate with the remediation host configured to provide data usable to remedy the insecure condition.

Ex. 2 at 22:35–23:9.

64. Claim 19 of the '705 Patent and its preamble and limitations are addressed above in Paragraphs 47-56.

65. The limitations of claim 17 of the '048 Patent are identical in many, if not most, respects to the limitations of Claim 19 of the '705 Patent.

66. The preamble and limitations [A] through [D] of the '048 Patent are met for the same reasons that the preamble and limitations [A] through [D] of the '705 Patent are met.

67. The preambles and the identical, common limitations of the Asserted Patents (e.g., limitations [A] through [D]) are addressed above, and Paragraphs 48-53 above are therefore incorporated herein by reference with respect to Claim 17 of the '048 Patent.

68. Limitations [E1] and [E2] of the '705 Patent differ in some respects from limitations [E1] and [E2] of the '048 Patent, as seen above.

69. Limitation [E1] of claim 17 the '048 Patent provides for "receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request[.]"

70. Limitation [E1] of claim 17 of the '048 Patent is met by the Accused Zero Trust Network Security Instrumentalities, as demonstrated (for example) by the publicly-available information below:

Use remediation messages and custom messages to help your users unblock themselves

When blocked, your users can encounter:

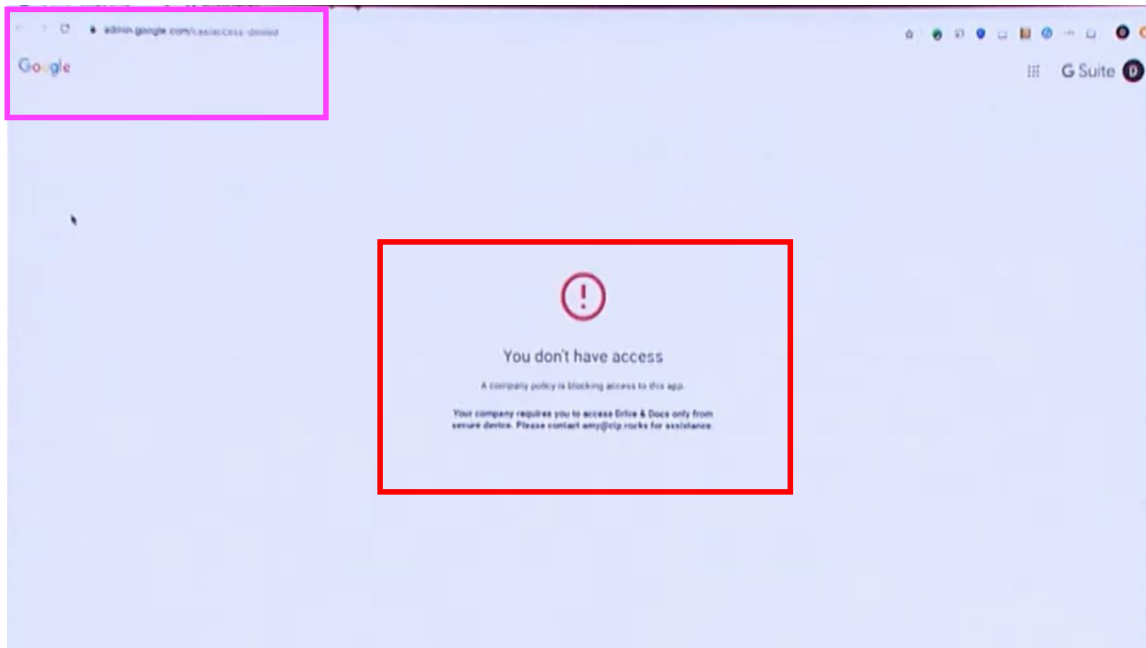
- **Default message**—Displays if you have not added remediation messages or custom messages. An example default message is: **Your organization's policy is blocking access to this app.**
- **Remediation messages**—Replaces the default message. The messages are system generated, and correspond to the specific policy violation that blocked the user. Remediation messages can present several remediation options to the user, which they can expand by clicking **Show more options**. In the case of several remediation options, the user needs to complete the steps for any one of the available options to unblock themselves.
- **Custom message**—Adds specific help for the user, such as additional advice on getting unblocked or a helpful link to click. You add custom messages as needed. A custom message can appear in conjunction with the default message, or with remediation messages.

See Ex. 12 at 1.

This page shows you how to enable and use the Chrome Enterprise Premium Policy Remediator.

When users attempt to access a Google Cloud resource but aren't compliant with the access policy for the resource, they are denied access and receive a general 403 error message. You can use the Policy Remediator to provide users with actionable steps that they can take to remediate their issue before reaching out to an admin for additional help. The specific remediation actions depend on the access policies, but can include things such as enabling screen lock, updating the operating system (OS) version, or accessing an app from a network allowed by your company.

See Ex. 13, Remediate Denied Access With the Policy Remediator, p. 1 (available at https://cloud.google.com/beyondcorp-enterprise/docs/policy-remediator?utm_source=chatgpt.com) (last accessed mid Jan. 2025 and incorporated by reference).

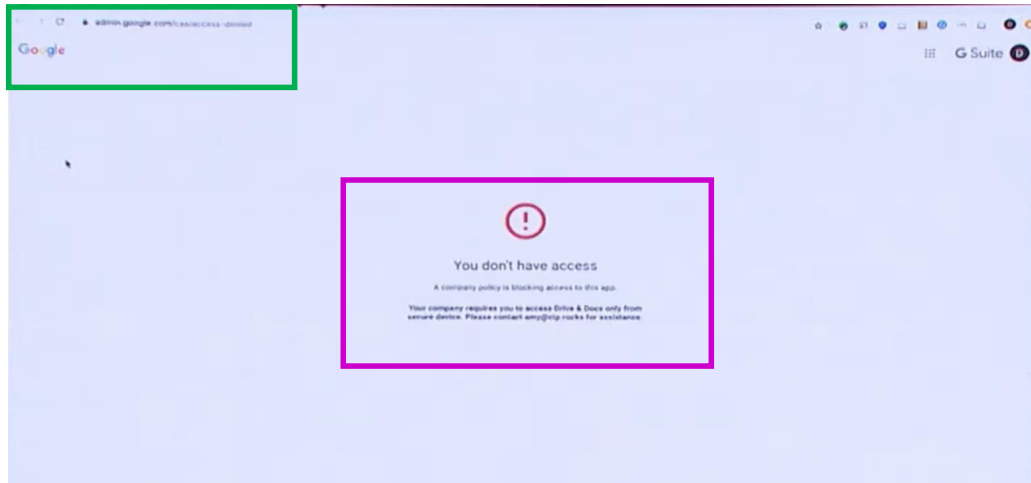


See Ex. 14, Video entitled "Enable Zero Trust Access Controls for your Web Apps, Infrastructure, and APIs (available at <https://www.youtube.com/watch?v=2jz5FQpGWQ4> at approx. 13:22) (last accessed mid Jan. 2025 and incorporated by reference).

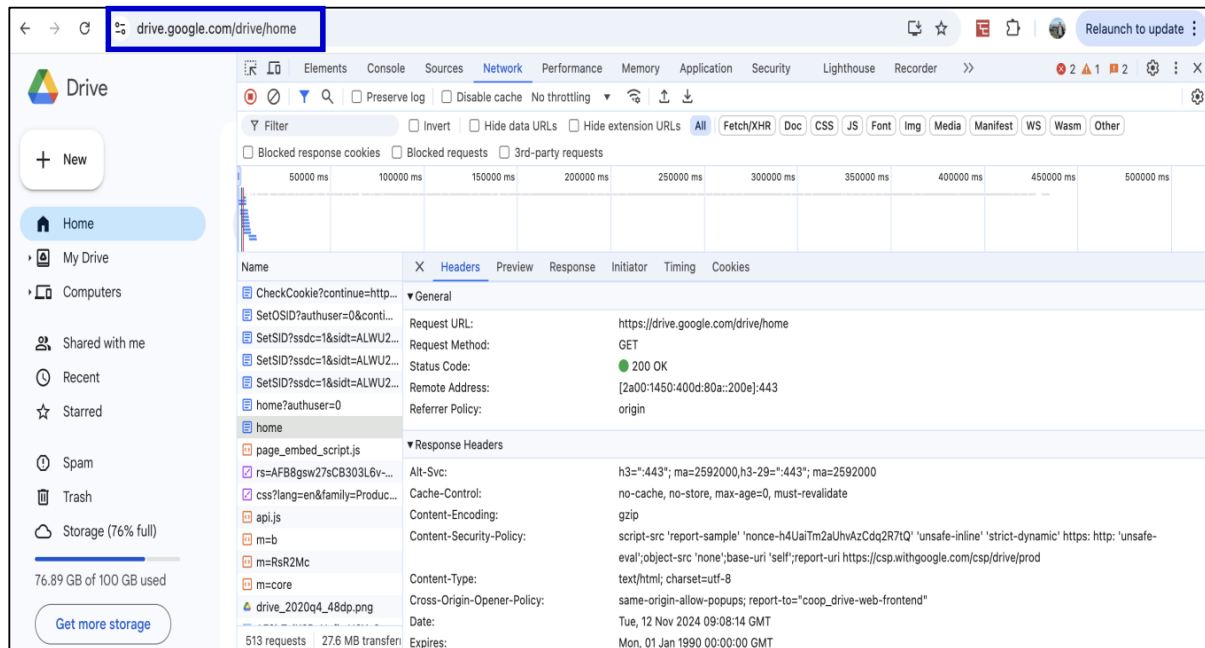
71. The Accused Zero Trust Network Security Instrumentalities determine that when a device (e.g., a first host computer) is non-compliant, the device is placed into a quarantined zone. The quarantined device may be allowed to make web service requests associated with remediation (e.g., a helpful link to click) while blocking service request associated with accessing protected network resources, and the Accused Zero Trust Network Security Instrumentalities can determine whether a request sent by the first host is associated with a remediation request.

72. Limitation [E2] of claim 17 of the '048 Patent provides that "wherein serving the quarantine notification page to the first host include re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page[.]"

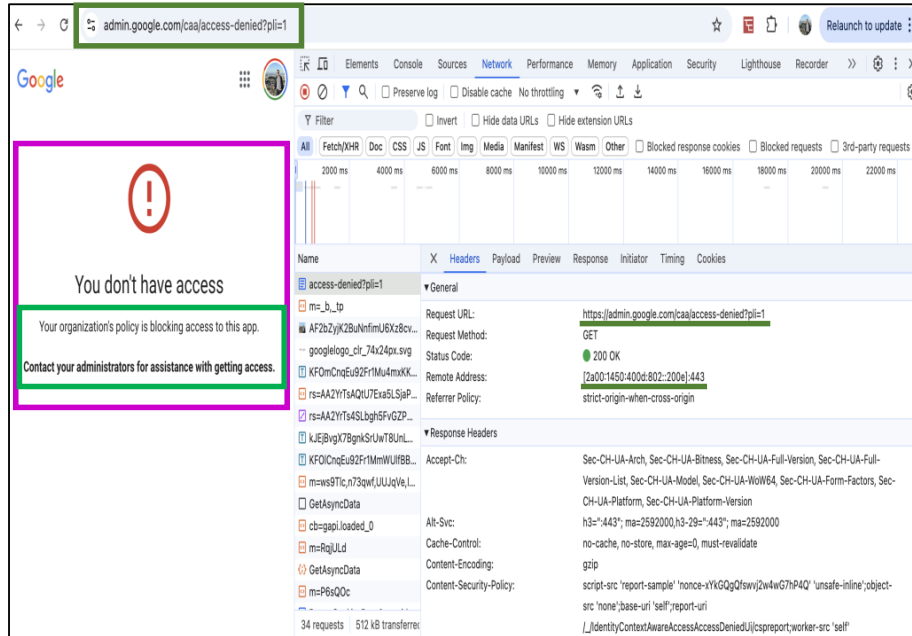
73. Limitation [E2] of claim 1 of the '048 Patent is met by the Accused Zero Trust Network Security Instrumentalities, as demonstrated (for example) by the publicly-available information below:



See Id.



See Ex. 15, Screenshot of page at <https://drive.google.com/drive/home> (last accessed early Dec. 2024 and incorporated by reference).



See Ex. 16, Screenshot of page at <https://admin.google.com/caa/access-denied?pli=1> (last accessed early Dec. 2024 and incorporated by reference).

74. As seen above, a user can be blocked from accessing network resources. In such an instance, a notification page stating "You don't have access," accompanied by a remediation message, is displayed. A remediation message and guidance on how to bring the device into compliance is provided. As depicted above, serving a quarantine page to a first host computer includes re-routing by responding to a service request with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve a quarantine notification page.

75. Limitation [F] of claim 17 of the '048 Patent is met for the same reasons as described in Paragraph 56 above (addressing limitation [F] of claim 19 of the '705 Patent).

76. Accordingly, the Accused Zero Trust Network Security Instrumentalities meet all the limitations of and therefore infringe at least Claim 17 of the '048 Patent.

77. Google's acts of infringement have occurred within this District and elsewhere throughout the United States.

78. As a result of Google's infringing conduct, K.Mizra has suffered damages. Google is liable to K.Mizra in an amount that adequately compensates K.Mizra for Google's infringement in an amount that is no less than a to be calculated fully paid-up, lump-sum, reasonable royalty, together with interest and costs as fixed by this Court under 25 U.S.C. § 284.

VII. REQUEST FOR RELIEF

WHEREFORE, K.Mizra respectfully requests the Court find in its favor and against Google, and that the Court grant K.Mizra at least the following relief:

A. Judgment that Google has infringed, literally and/or under the doctrine of equivalents, one or more claims of the Asserted Patents;

B. Awarding damages to K.Mizra in an amount to be proven at trial and in the form of a fully paid-up, lump sum, reasonable royalty that takes into account and runs through expiration of the Asserted Patents;

C. Awarding enhanced damages, as appropriate, under 35 U.S.C. § 284;

D. Awarding K.Mizra's costs (including internal and external costs and disbursements) and declaring this an exceptional case and awarding K.Mizra its attorneys' fees in accordance with 35 U.S.C. § 285;

E. Pre-judgment and post-judgment interest at the maximum rate permitted by law on the damages caused by reason of Google's infringing activities and other conduct complained of herein; and

F. Awarding such other and further relief as this Court deems just and proper under the circumstances.

VIII. DEMAND FOR JURY TRIAL

Pursuant to Fed. R. Civ. P. 38(b), K.Mizra hereby demands a trial by jury on all issues so triable.

Dated: February 18, 2025

Respectfully submitted,

By: /s/ Michael C. Smith

Michael C. Smith
Texas Bar No. 18650410
michael.smith@solidcounsel.com
Scheef & Stone, LLP
113 E. Austin Street
Marshall, TX 75670
(903) 938-8900

Robert R. Brunelli*
CO State Bar No. 20070
rbunelli@sheridanross.com

Scott R. Bialecki*
CO State Bar No. 23103
sbialecki@sheridanross.com

Bart A. Starr*
CO State Bar No. 50446
bstarr@sheridanross.com

Brian S. Boerman*
CO State Bar No. 50834
bboerman@sheridanross.com

Gene Volchenko*
IL State Bar No. 6342818
gvolchenko@sheridanross.com

SHERIDAN ROSS P.C.
1560 Broadway, Suite 1200
Denver, CO 80202
Telephone: 303-863-9700
Facsimile: 303-863-0223
litigation@sheridanross.com

Of Counsel:

Claire A. Henry
Texas Bar No. 24053063
Miller Fair Henry PLLC
1507 Bill Owens Parkway
Longview, TX 75604
Telephone: (903) 757-6400

Facsimile: (903) 757-2323
claire@millerfairhenry.com

**Pro hac vice pending*
Attorneys for Plaintiff K.Mizra LLC

CIVIL COVER SHEET

The JS 44 civil cover sheet and the information contained herein neither replace nor supplement the filing and service of pleadings or other papers as required by law, except as provided by local rules of court. This form, approved by the Judicial Conference of the United States in September 1974, is required for the use of the Clerk of Court for the purpose of initiating the civil docket sheet. (SEE INSTRUCTIONS ON NEXT PAGE OF THIS FORM.)

I. (a) PLAINTIFFS K.Mizra LLC (b) County of Residence of First Listed Plaintiff <u>Miami-Dade, FL</u> (EXCEPT IN U.S. PLAINTIFF CASES) (c) Attorneys (Firm Name, Address, and Telephone Number) Michael C. Smith, Scheef & Stone, LLP, 113 E. Austin Street, Marshall, TX 75670 (903-938-8900)	DEFENDANTS Google LLC County of Residence of First Listed Defendant _____ (IN U.S. PLAINTIFF CASES ONLY) NOTE: IN LAND CONDEMNATION CASES, USE THE LOCATION OF THE TRACT OF LAND INVOLVED. Attorneys (If Known)
--	---

II. BASIS OF JURISDICTION (Place an "X" in One Box Only) ☐ 1 U.S. Government Plaintiff ☒ 3 Federal Question (U.S. Government Not a Party) ☐ 2 U.S. Government Defendant ☐ 4 Diversity (Indicate Citizenship of Parties in Item III)	III. CITIZENSHIP OF PRINCIPAL PARTIES (Place an "X" in One Box for Plaintiff and One Box for Defendant) <table style="width: 100%;"><tr><th></th><th>PTF</th><th>DEF</th><th></th><th>PTF</th><th>DEF</th></tr><tr><td>Citizen of This State</td><td>☐ 1</td><td>☐ 1</td><td>Incorporated or Principal Place of Business In This State</td><td>☐ 4</td><td>☐ 4</td></tr><tr><td>Citizen of Another State</td><td>☐ 2</td><td>☐ 2</td><td>Incorporated and Principal Place of Business In Another State</td><td>☐ 5</td><td>☐ 5</td></tr><tr><td>Citizen or Subject of a Foreign Country</td><td>☐ 3</td><td>☐ 3</td><td>Foreign Nation</td><td>☐ 6</td><td>☐ 6</td></tr></table>		PTF	DEF		PTF	DEF	Citizen of This State	☐ 1	☐ 1	Incorporated or Principal Place of Business In This State	☐ 4	☐ 4	Citizen of Another State	☐ 2	☐ 2	Incorporated and Principal Place of Business In Another State	☐ 5	☐ 5	Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6
	PTF	DEF		PTF	DEF																				
Citizen of This State	☐ 1	☐ 1	Incorporated or Principal Place of Business In This State	☐ 4	☐ 4																				
Citizen of Another State	☐ 2	☐ 2	Incorporated and Principal Place of Business In Another State	☐ 5	☐ 5																				
Citizen or Subject of a Foreign Country	☐ 3	☐ 3	Foreign Nation	☐ 6	☐ 6																				

IV. NATURE OF SUIT (Place an "X" in One Box Only)			Click here for: Nature of Suit Code Descriptions.	
CONTRACT ☐ 110 Insurance ☐ 120 Marine ☐ 130 Miller Act ☐ 140 Negotiable Instrument ☐ 150 Recovery of Overpayment & Enforcement of Judgment ☐ 151 Medicare Act ☐ 152 Recovery of Defaulted Student Loans (Excludes Veterans) ☐ 153 Recovery of Overpayment of Veteran's Benefits ☐ 160 Stockholders' Suits ☐ 190 Other Contract ☐ 195 Contract Product Liability ☐ 196 Franchise	TORTS PERSONAL INJURY ☐ 310 Airplane ☐ 315 Airplane Product Liability ☐ 320 Assault, Libel & Slander ☐ 330 Federal Employers' Liability ☐ 340 Marine ☐ 345 Marine Product Liability ☐ 350 Motor Vehicle ☐ 355 Motor Vehicle Product Liability ☐ 360 Other Personal Injury ☐ 362 Personal Injury - Medical Malpractice PERSONAL INJURY ☐ 365 Personal Injury - Product Liability ☐ 367 Health Care/Pharmaceutical Personal Injury Product Liability PERSONAL PROPERTY ☐ 370 Other Fraud ☐ 371 Truth in Lending ☐ 380 Other Personal Property Damage ☐ 385 Property Damage Product Liability	FORFEITURE/PENALTY ☐ 625 Drug Related Seizure of Property 21 USC 881 ☐ 690 Other LABOR ☐ 710 Fair Labor Standards Act ☐ 720 Labor/Management Relations ☐ 740 Railway Labor Act ☐ 751 Family and Medical Leave Act ☐ 790 Other Labor Litigation ☐ 791 Employee Retirement Income Security Act IMMIGRATION ☐ 462 Naturalization Application ☐ 465 Other Immigration Actions	BANKRUPTCY ☐ 422 Appeal 28 USC 158 ☐ 423 Withdrawal 28 USC 157 PROPERTY RIGHTS ☐ 820 Copyrights ☒ 830 Patent ☐ 835 Patent - Abbreviated New Drug Application ☐ 840 Trademark SOCIAL SECURITY ☐ 861 HIA (1395ff) ☐ 862 Black Lung (923) ☐ 863 DIWC/DIWW (405(g)) ☐ 864 SSID Title XVI ☐ 865 RSI (405(g)) FEDERAL TAX SUITS ☐ 870 Taxes (U.S. Plaintiff or Defendant) ☐ 871 IRS—Third Party 26 USC 7609	OTHER STATUTES ☐ 375 False Claims Act ☐ 376 Qui Tam (31 USC 3729(a)) ☐ 400 State Reapportionment ☐ 410 Antitrust ☐ 430 Banks and Banking ☐ 450 Commerce ☐ 460 Deportation ☐ 470 Racketeer Influenced and Corrupt Organizations ☐ 480 Consumer Credit (15 USC 1681 or 1692) ☐ 485 Telephone Consumer Protection Act ☐ 490 Cable/Sat TV ☐ 850 Securities/Commodities/Exchange ☐ 890 Other Statutory Actions ☐ 891 Agricultural Acts ☐ 893 Environmental Matters ☐ 895 Freedom of Information Act ☐ 896 Arbitration ☐ 899 Administrative Procedure Act/Review or Appeal of Agency Decision ☐ 950 Constitutionality of State Statutes

V. ORIGIN (Place an "X" in One Box Only)							
☒ 1 Original Proceeding	☐ 2 Removed from State Court	☐ 3 Remanded from Appellate Court	☐ 4 Reinstated or Reopened	☐ 5 Transferred from Another District (specify)	☐ 6 Multidistrict Litigation - Transfer	☐ 8 Multidistrict Litigation - Direct File	

VI. CAUSE OF ACTION	Cite the U.S. Civil Statute under which you are filing (Do not cite jurisdictional statutes unless diversity): <u>35 U.S.C. § 1 et seq</u> Brief description of cause: <u>Patent infringement</u>
----------------------------	--

VII. REQUESTED IN COMPLAINT:	☐ CHECK IF THIS IS A CLASS ACTION UNDER RULE 23, F.R.Cv.P.	DEMAND \$	CHECK YES only if demanded in complaint: JURY DEMAND: ☒ Yes ☐ No
-------------------------------------	---	------------------	---

VIII. RELATED CASE(S) IF ANY	(See instructions): JUDGE <u>Judge Alan D Albright</u> DOCKET NUMBER <u>6:20-cv-1031-ADA</u>
-------------------------------------	--

DATE <u>02/18/2025</u>	SIGNATURE OF ATTORNEY OF RECORD <u>/s/ Michael C. Smith</u>
---------------------------	--

RECEIPT #	AMOUNT	APPLYING IFP	JUDGE	MAG. JUDGE
-----------	--------	--------------	-------	------------

K.Mizra LLC, Ex. 2019, p. 37
Google LLC v. K.Mizra LLC
IPR2025-01436

EXHIBIT 1



US008234705B1

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

(10) **Patent No.:** **US 8,234,705 B1**

(45) **Date of Patent:** **Jul. 31, 2012**

(54) **CONTAGION ISOLATION AND INOCULATION**

(75) Inventors: **James A. Roskind**, Redwood City, CA (US); **Aaron T. Emigh**, Incline Village, NV (US)

(73) Assignee: **Radix Holdings, LLC**, Incline Village, NV (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 1520 days.

(21) Appl. No.: **11/237,003**

(22) Filed: **Sep. 27, 2005**

Related U.S. Application Data

(60) Provisional application No. 60/613,909, filed on Sep. 27, 2004.

(51) **Int. Cl.**
G06F 11/00 (2006.01)
G06F 12/14 (2006.01)
G06F 12/16 (2006.01)
G08B 23/00 (2006.01)

(52) **U.S. Cl.** **726/22; 726/23; 726/24; 726/25**

(58) **Field of Classification Search** **726/11, 726/14, 22-25**

See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,823,479 B1 * 11/2004 McElhaney et al. 714/43
6,996,845 B1 * 2/2006 Hurst et al. 726/25
7,263,720 B2 * 8/2007 Silvester 726/22
7,287,278 B2 * 10/2007 Liang 726/22
7,424,610 B2 * 9/2008 Ranganathan 713/164
2002/0083328 A1 * 6/2002 Riordan 713/183
2003/0188173 A1 * 10/2003 Zimmer et al. 713/189

2003/0188194 A1 * 10/2003 Currie et al. 713/201
2005/0018618 A1 * 1/2005 Muallem et al. 370/252
2005/0033987 A1 * 2/2005 Yan et al. 713/201
2005/0131997 A1 * 6/2005 Lewis et al. 709/203
2006/0053426 A1 * 3/2006 Dive-Reclus et al. 719/328

OTHER PUBLICATIONS

Bohra, A.; Neamtiu, I.; Gallard, P.; Sultan, F.; Ifode, L.; Remote repair of operating system state using Backdoors; Autonomic Computing, 2004. Proceedings. International Conference on May 17-18, 2004 pp. 256-263.*

Mark V. Glover, Integrating a Trusted Computing Base Extension Server and Secure Session Server into the LINUX Operating System, Sep. 2001, Naval Postgraduate School.*

Pearson et al., Trusted Computing Platform: TCPA Technology in Context, 2003, Hewlett-Packard Books; Patricia Pekary.*

OLS: Linux and trusted computing, Jul. 22, 2005, <http://lwn.net/Articles/144681/> (as lined on Jul. 15, 2010).*

* cited by examiner

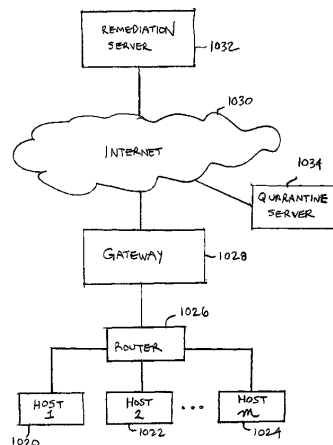
Primary Examiner — Eleni Shiferaw

Assistant Examiner — Jing Sims

(57) **ABSTRACT**

Contagion isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

19 Claims, 25 Drawing Sheets



Ex. 1, p. 1

K.Mizra LLC, Ex. 2019, p. 39

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 1 of 25

US 8,234,705 B1

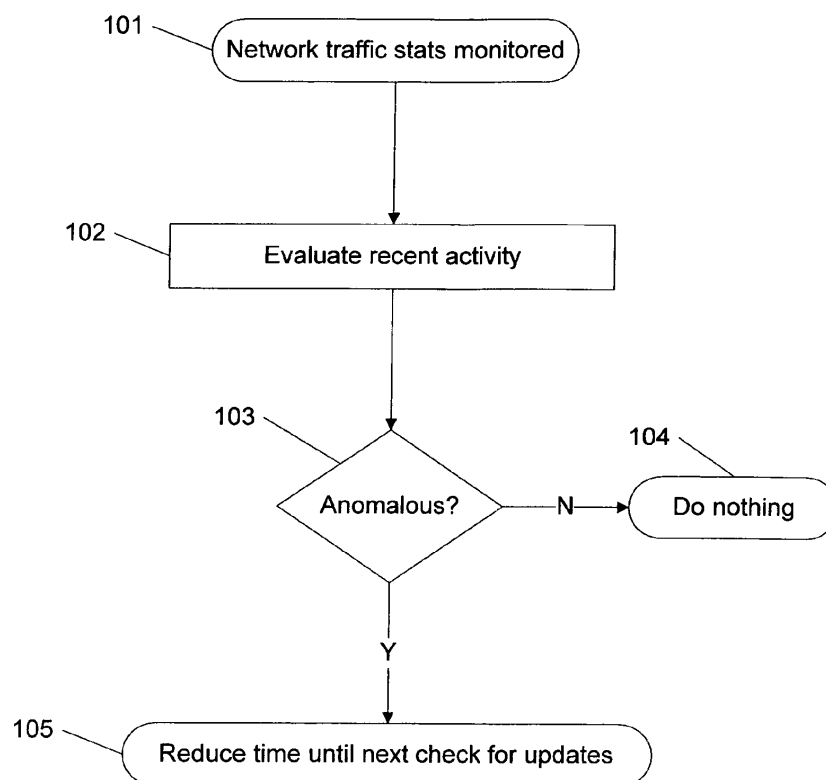


Figure 1

Ex. 1, p. 2

K.Mizra LLC, Ex. 2019, p. 40
Google LLC v. K.Mizra LLC
IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 2 of 25

US 8,234,705 B1

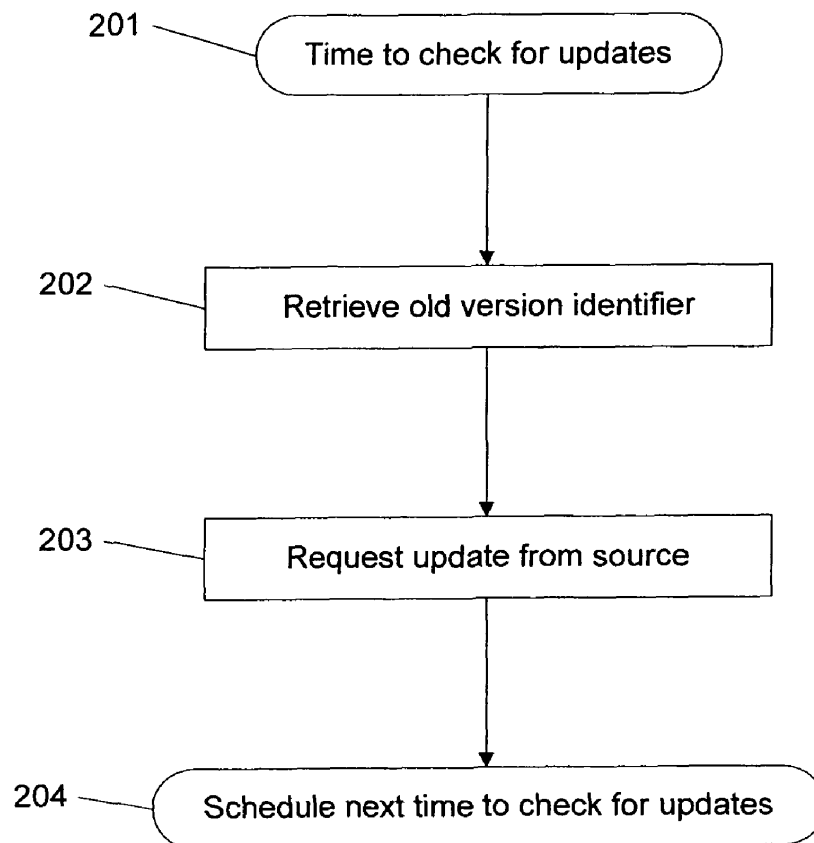


Figure 2A

Ex. 1, p. 3

K.Mizra LLC, Ex. 2019, p. 41
Google LLC v. K.Mizra LLC
IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 3 of 25

US 8,234,705 B1

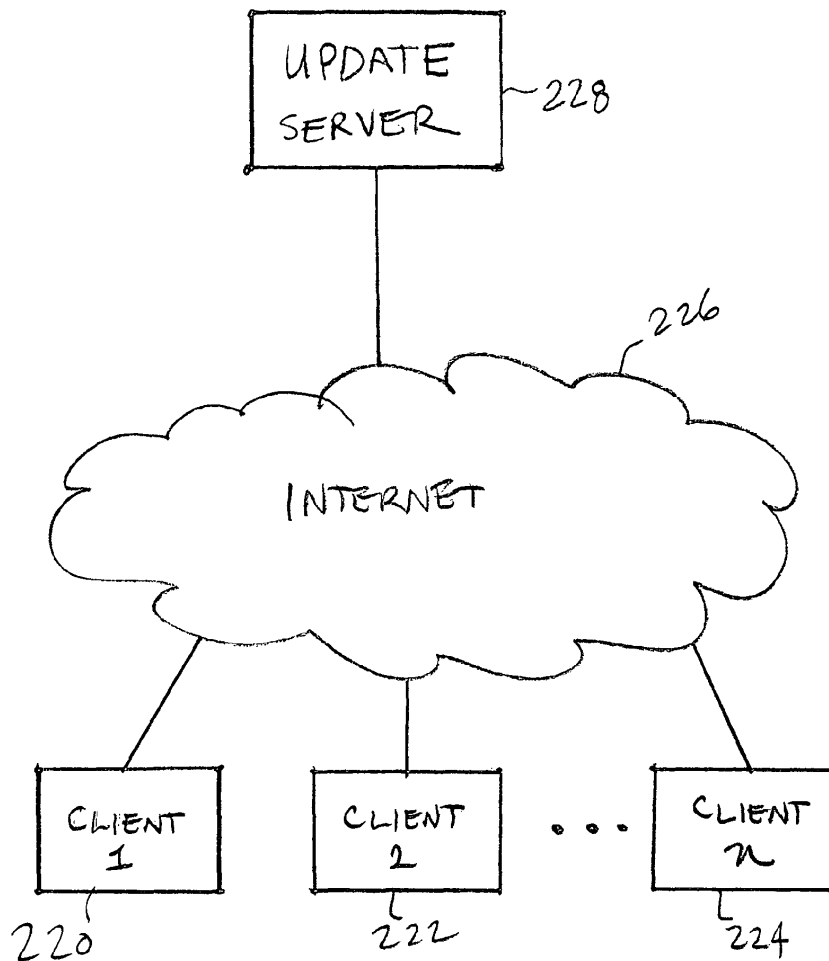


FIG. 2B

Ex. 1, p. 4

K.Mizra LLC, Ex. 2019, p. 42

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 4 of 25

US 8,234,705 B1

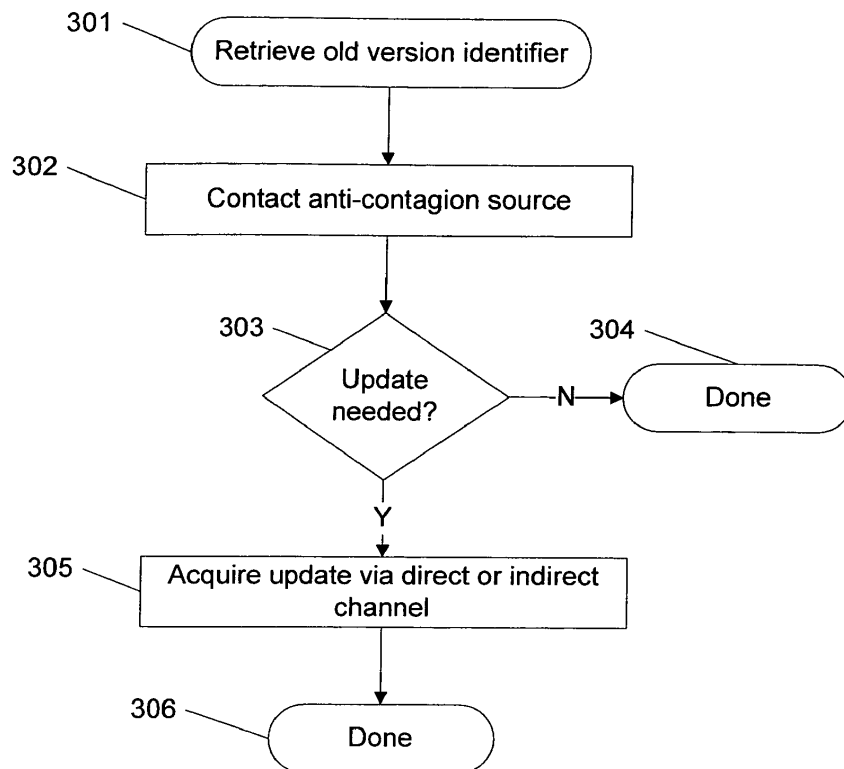


Figure 3A

Ex. 1, p. 5

K.Mizra LLC, Ex. 2019, p. 43

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 5 of 25

US 8,234,705 B1

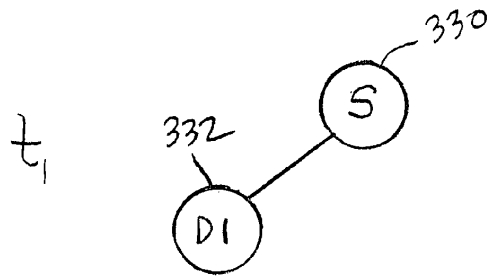


FIG. 3B

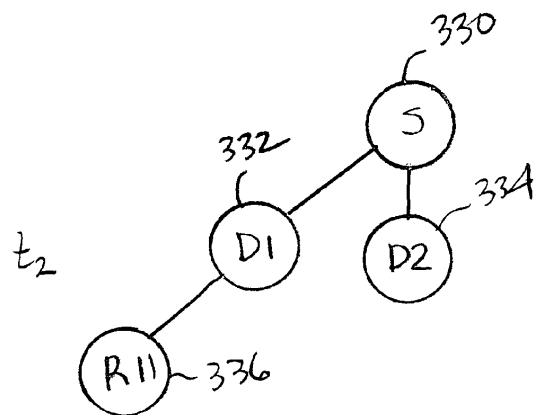


FIG. 3C

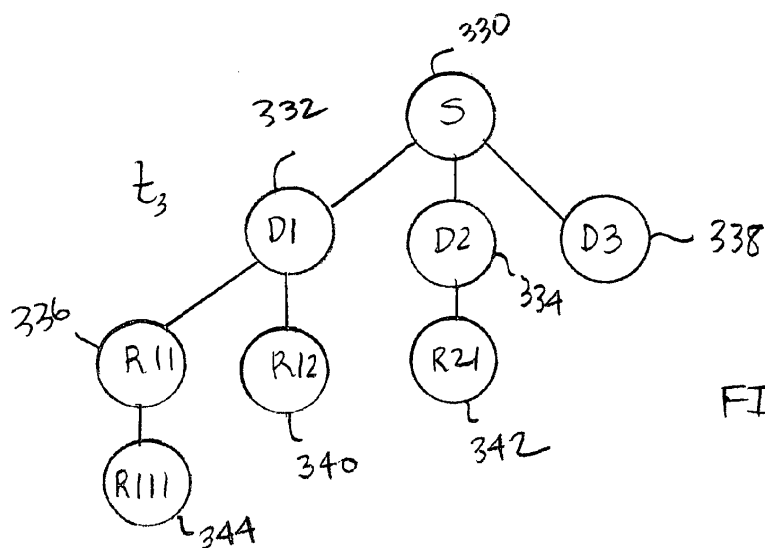


FIG. 3D

Ex. 1, p. 6

K.Mizra LLC, Ex. 2019, p. 44

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 6 of 25

US 8,234,705 B1

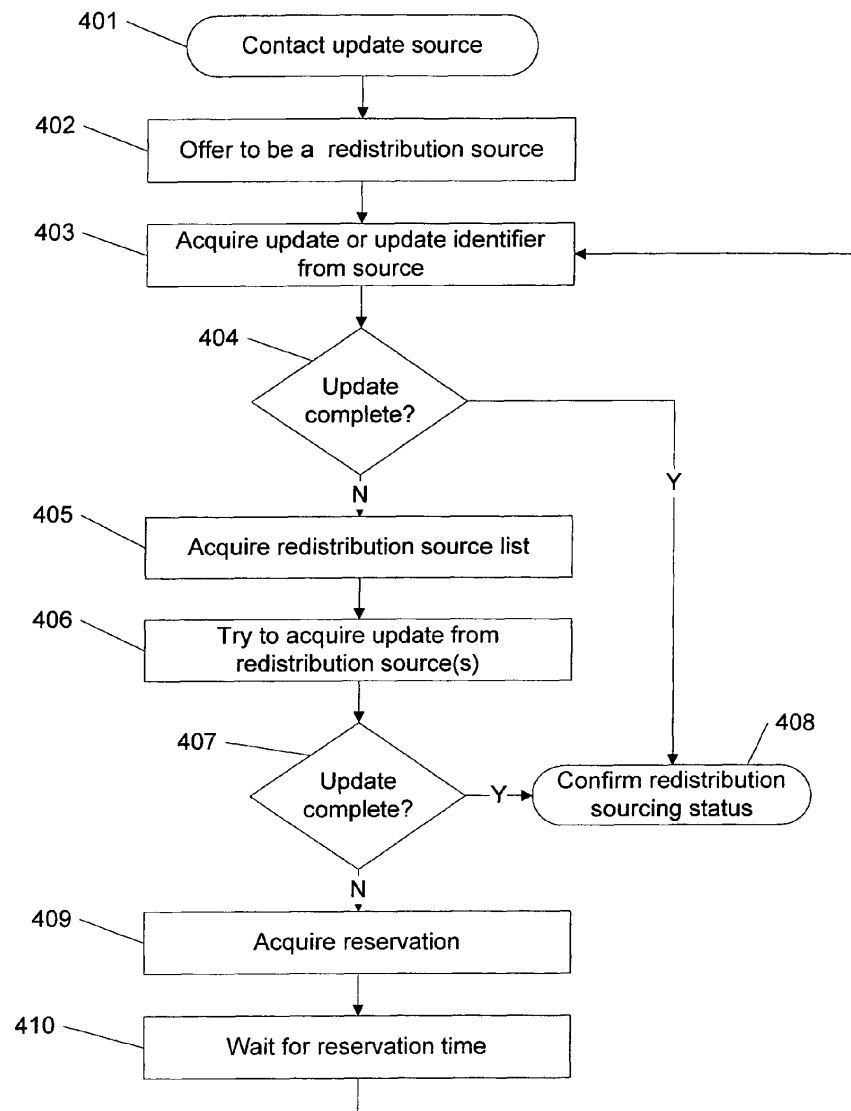


Figure 4A

Ex. 1, p. 7

K.Mizra LLC, Ex. 2019, p. 45

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 7 of 25

US 8,234,705 B1

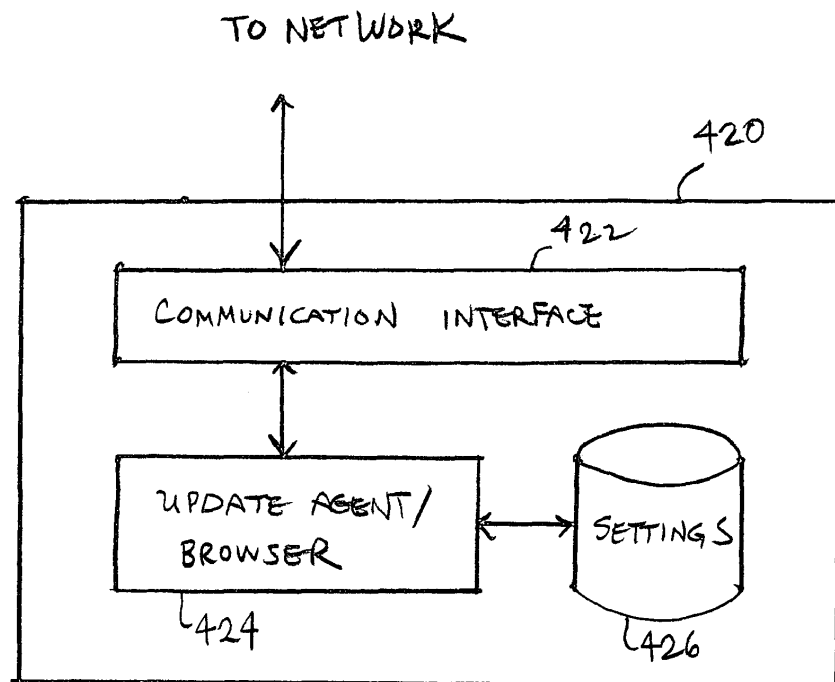


FIG. 4B

Ex. 1, p. 8

K.Mizra LLC, Ex. 2019, p. 46

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 8 of 25

US 8,234,705 B1

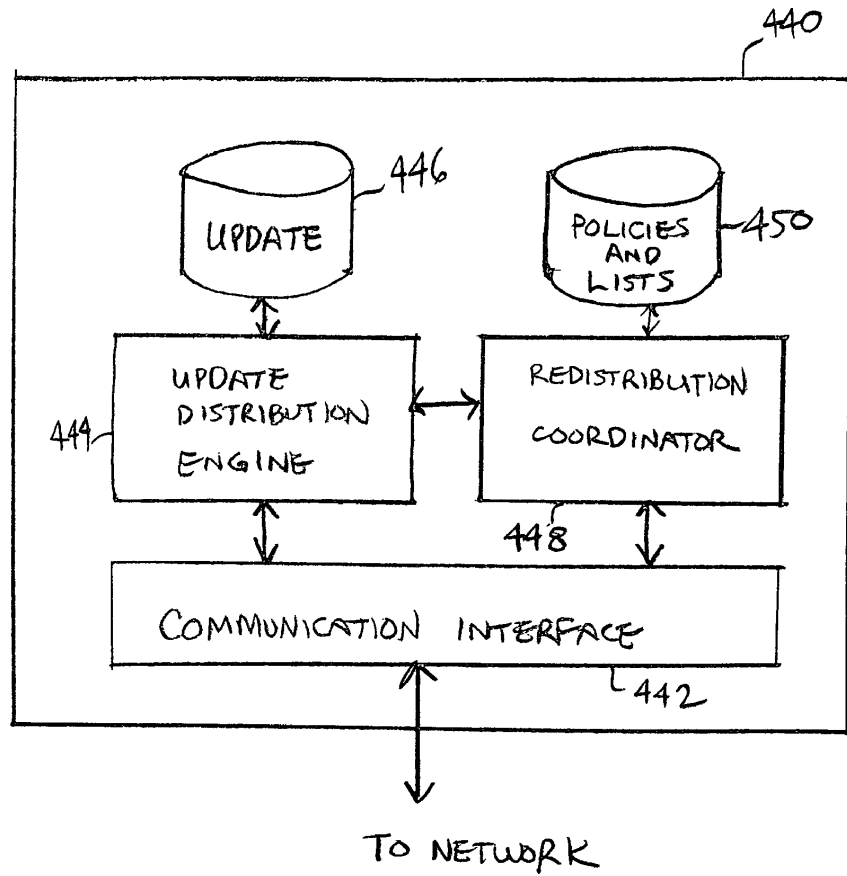


FIG. 4C

Ex. 1, p. 9

K.Mizra LLC, Ex. 2019, p. 47

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 9 of 25

US 8,234,705 B1

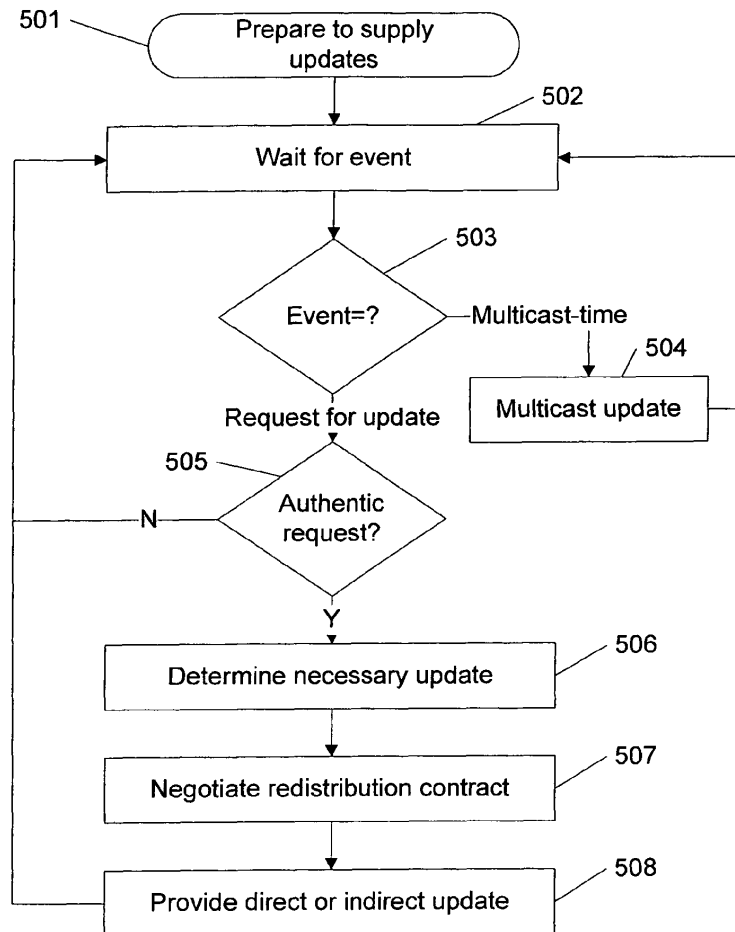


Figure 5

Ex. 1, p. 10

K.Mizra LLC, Ex. 2019, p. 48

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 10 of 25

US 8,234,705 B1

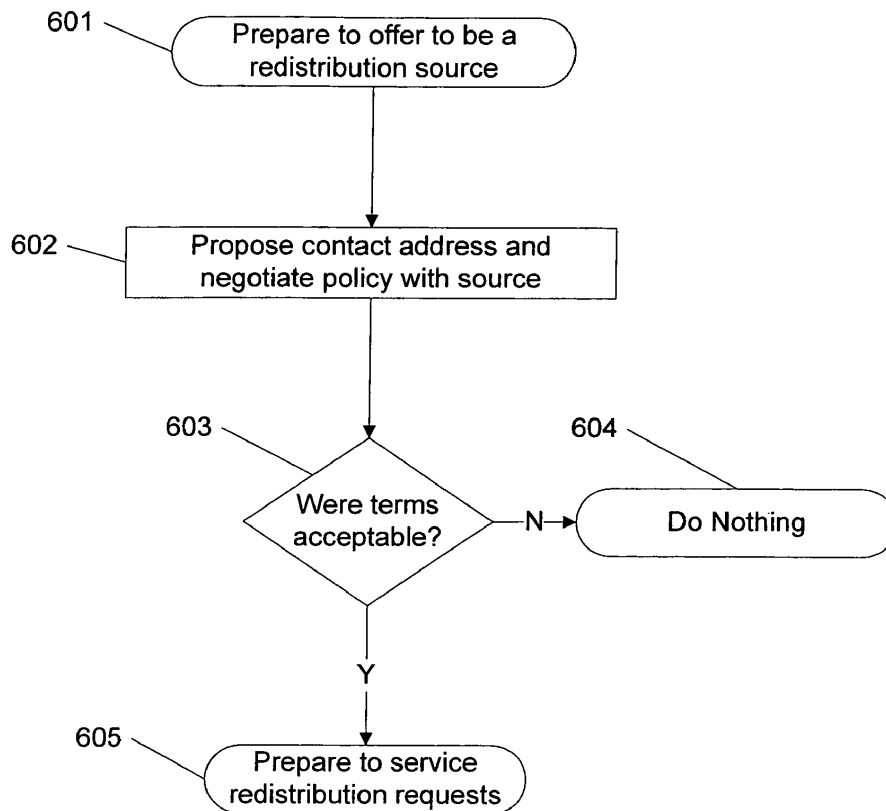


Figure 6

U.S. Patent

Jul. 31, 2012

Sheet 11 of 25

US 8,234,705 B1

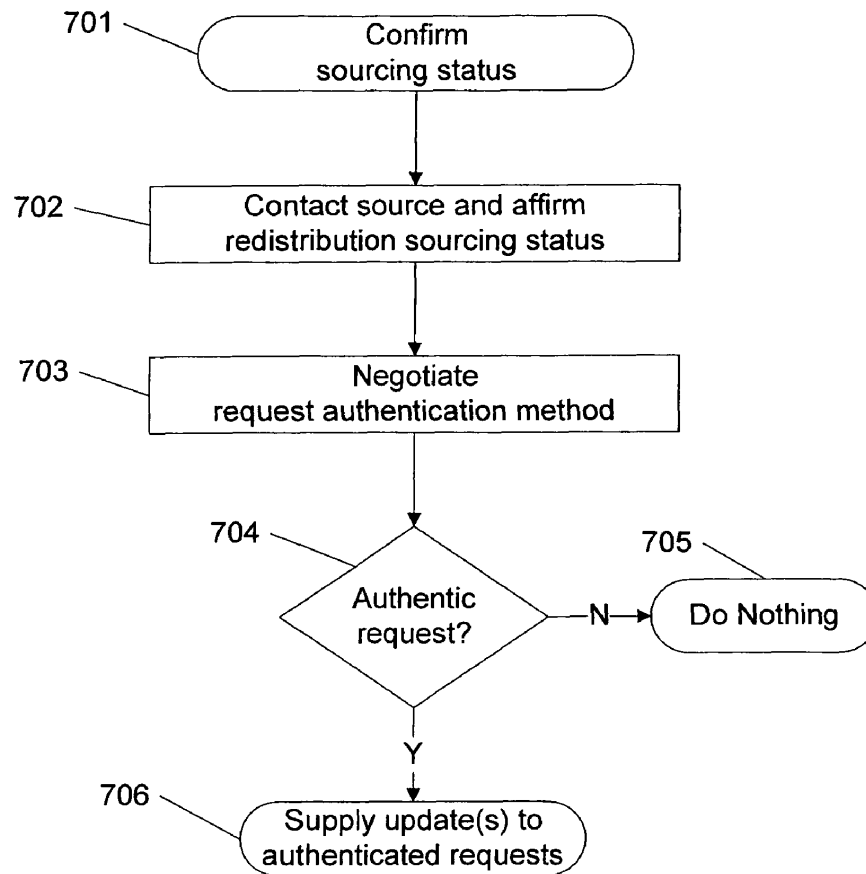


Figure 7

U.S. Patent

Jul. 31, 2012

Sheet 12 of 25

US 8,234,705 B1

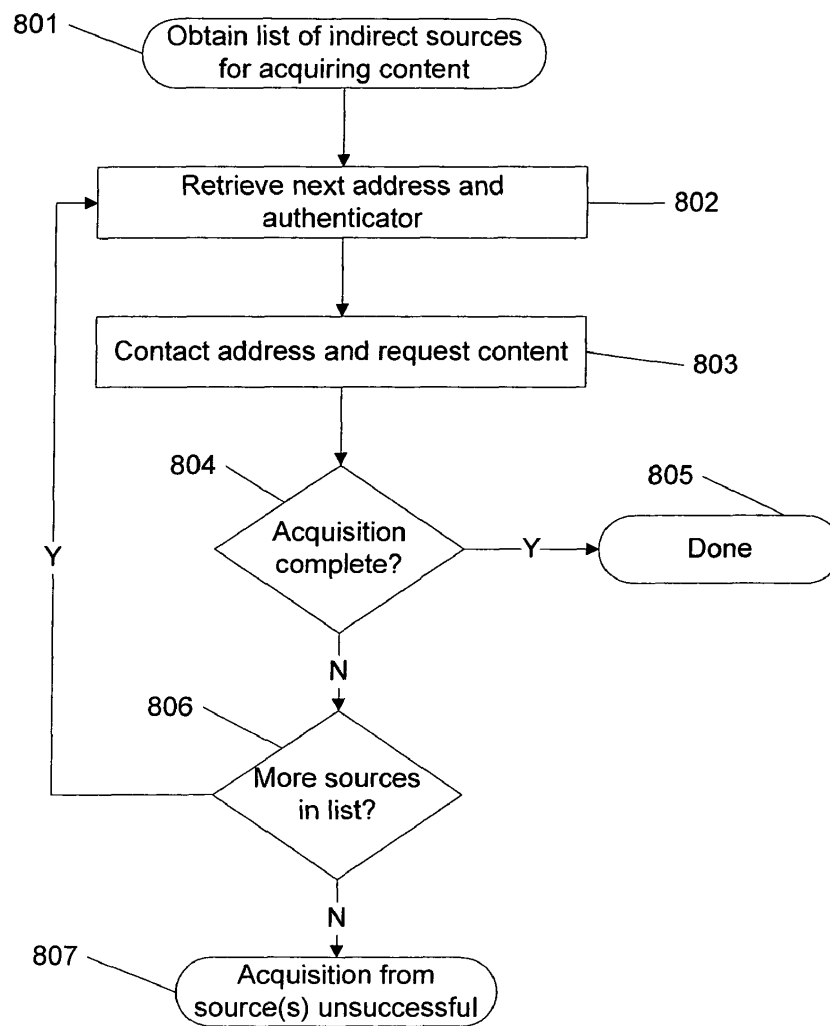


Figure 8

U.S. Patent

Jul. 31, 2012

Sheet 13 of 25

US 8,234,705 B1

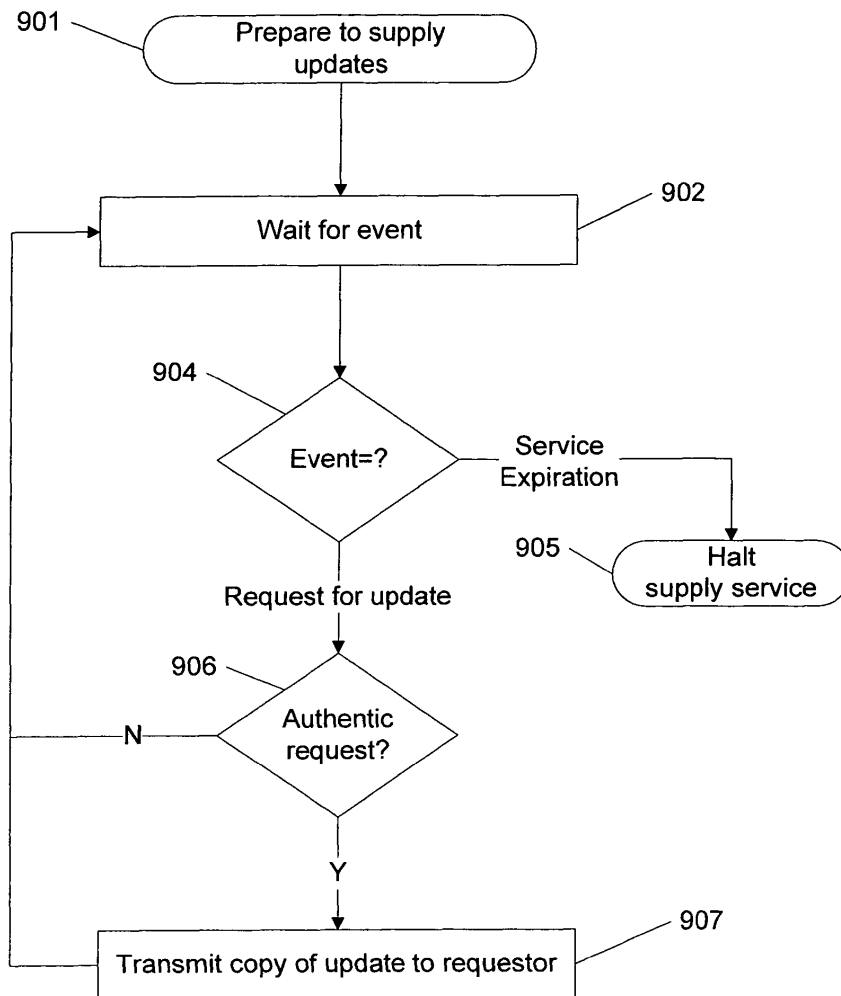


Figure 9

U.S. Patent

Jul. 31, 2012

Sheet 14 of 25

US 8,234,705 B1

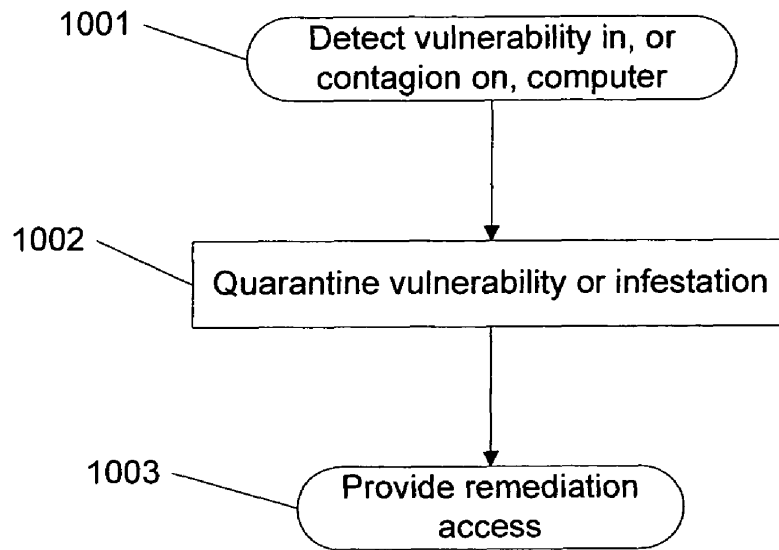


Figure 10A

Ex. 1, p. 15

K.Mizra LLC, Ex. 2019, p. 53

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 15 of 25

US 8,234,705 B1

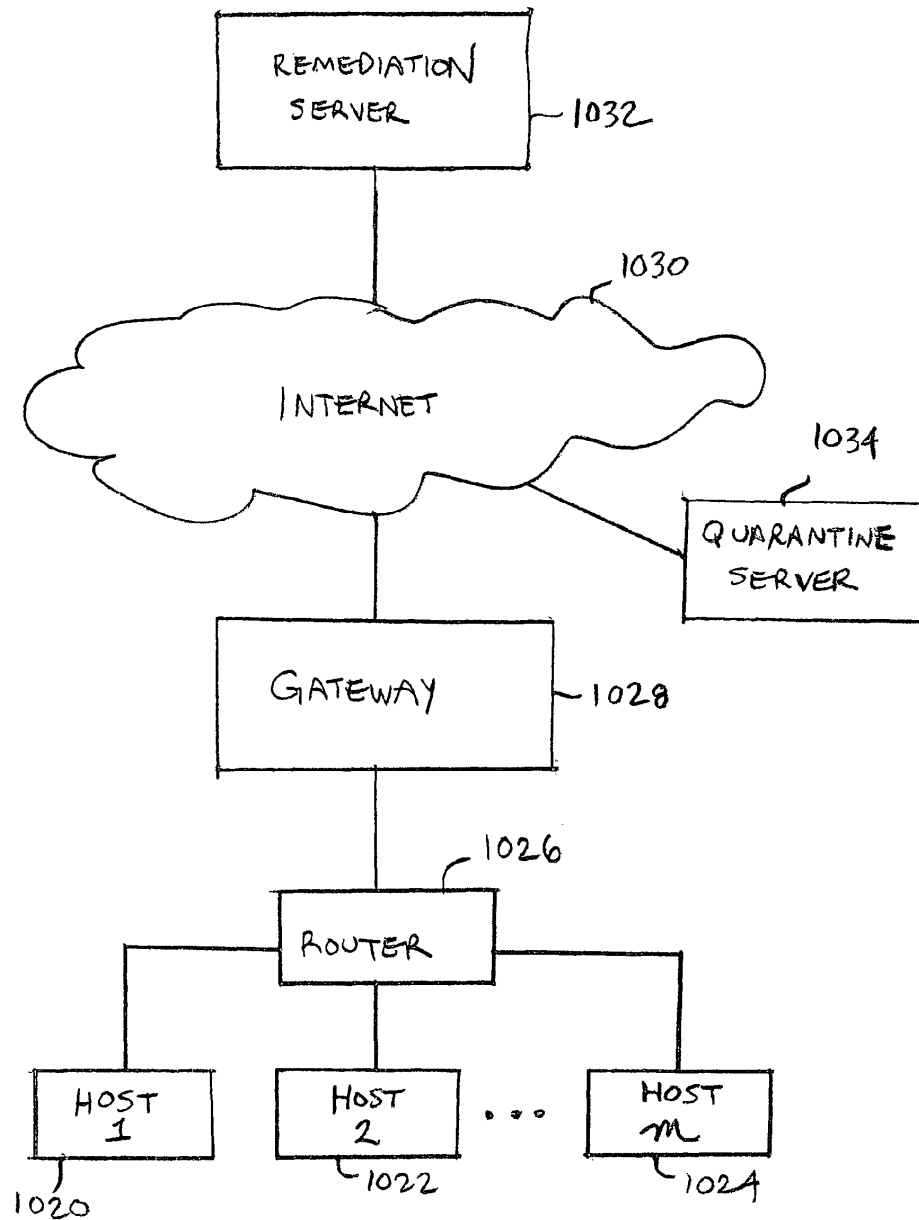


FIG. 10B

Ex. 1, p. 16

K.Mizra LLC, Ex. 2019, p. 54

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 16 of 25

US 8,234,705 B1

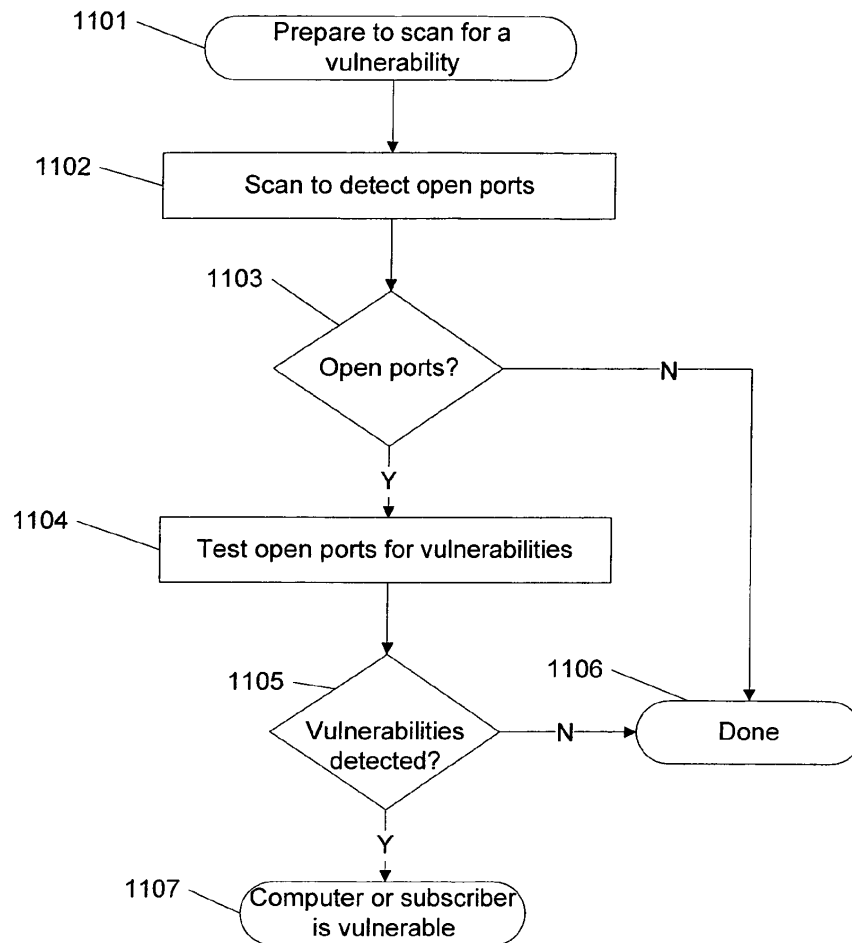


Figure 11

Ex. 1, p. 17

K.Mizra LLC, Ex. 2019, p. 55

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 17 of 25

US 8,234,705 B1

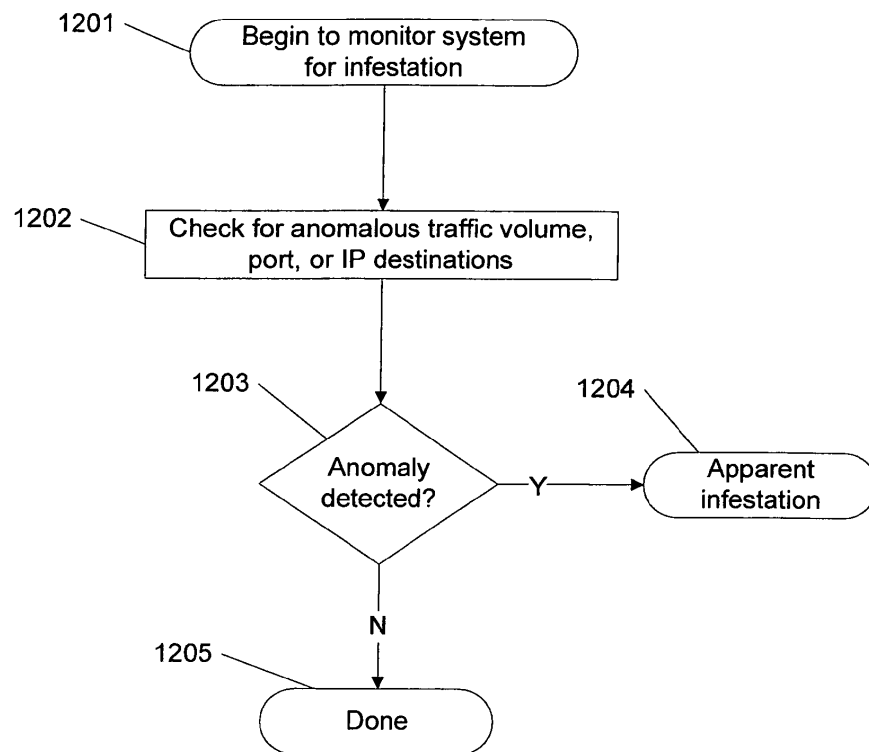


Figure 12

U.S. Patent

Jul. 31, 2012

Sheet 18 of 25

US 8,234,705 B1

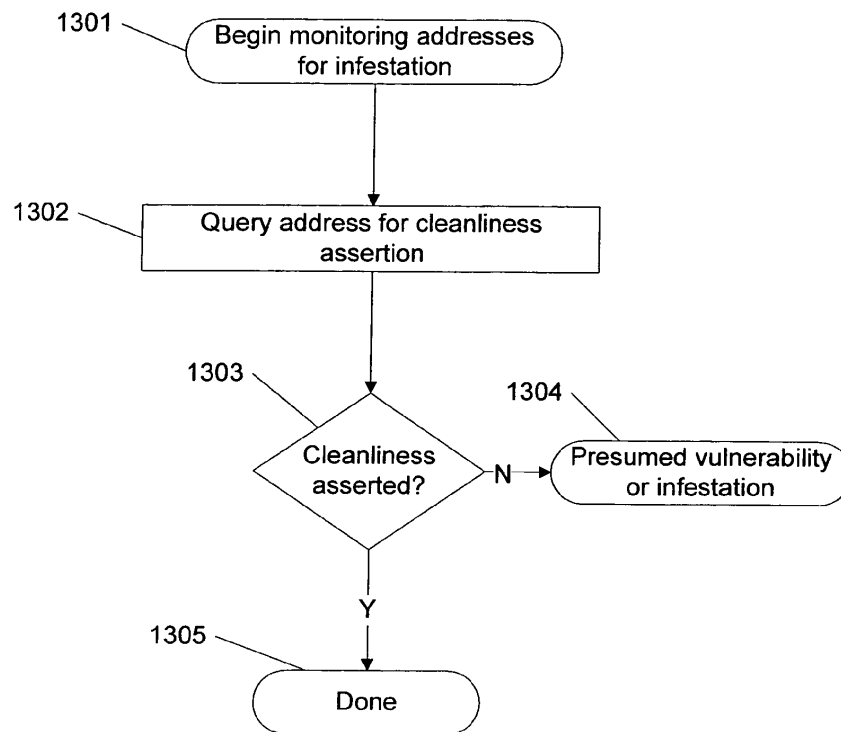


Figure 13

U.S. Patent

Jul. 31, 2012

Sheet 19 of 25

US 8,234,705 B1

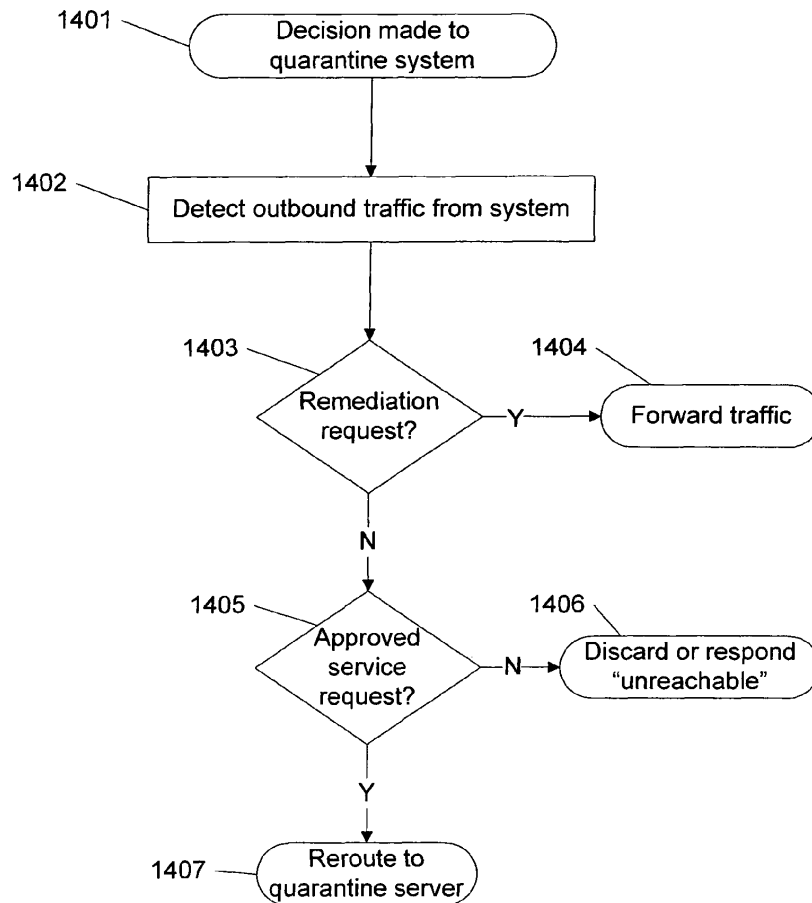


Figure 14

Ex. 1, p. 20

K.Mizra LLC, Ex. 2019, p. 58

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 20 of 25

US 8,234,705 B1

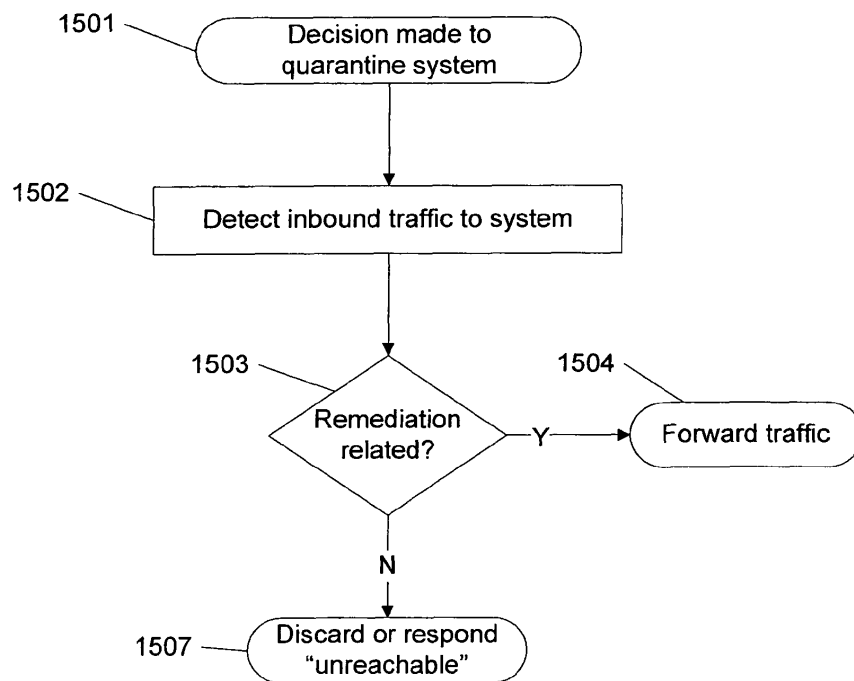


Figure 15

Ex. 1, p. 21

K.Mizra LLC, Ex. 2019, p. 59
Google LLC v. K.Mizra LLC
IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 21 of 25

US 8,234,705 B1

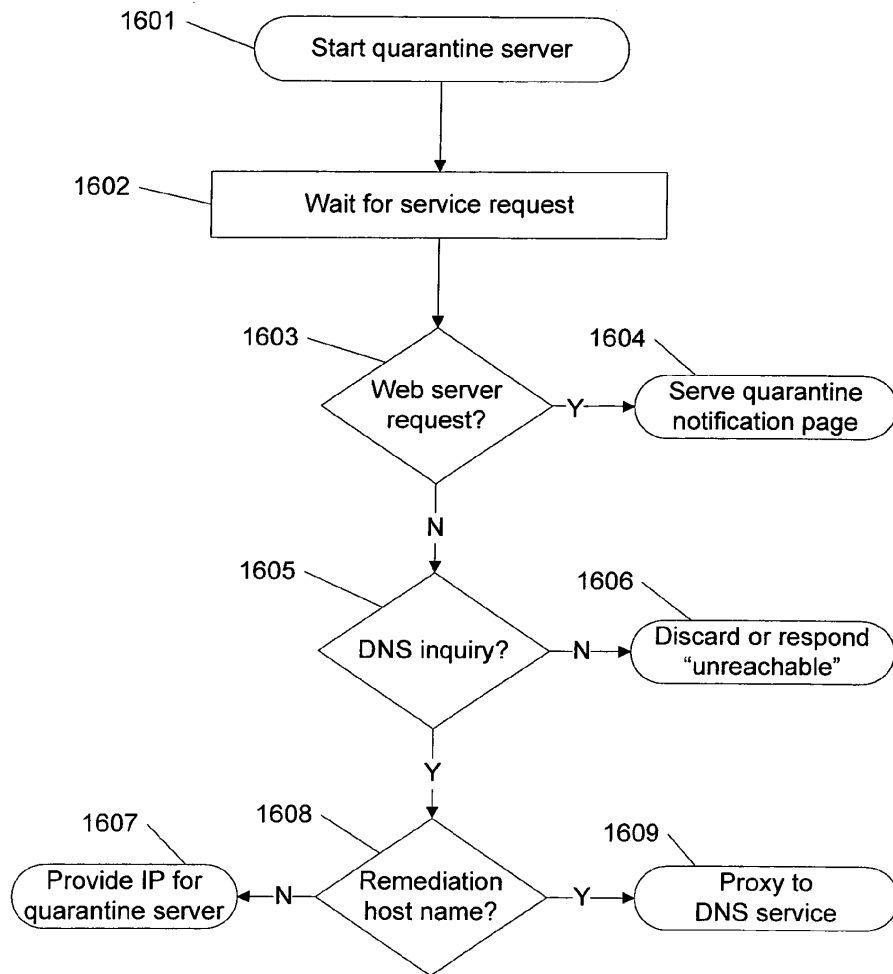


Figure 16

Ex. 1, p. 22

K.Mizra LLC, Ex. 2019, p. 60

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 22 of 25

US 8,234,705 B1

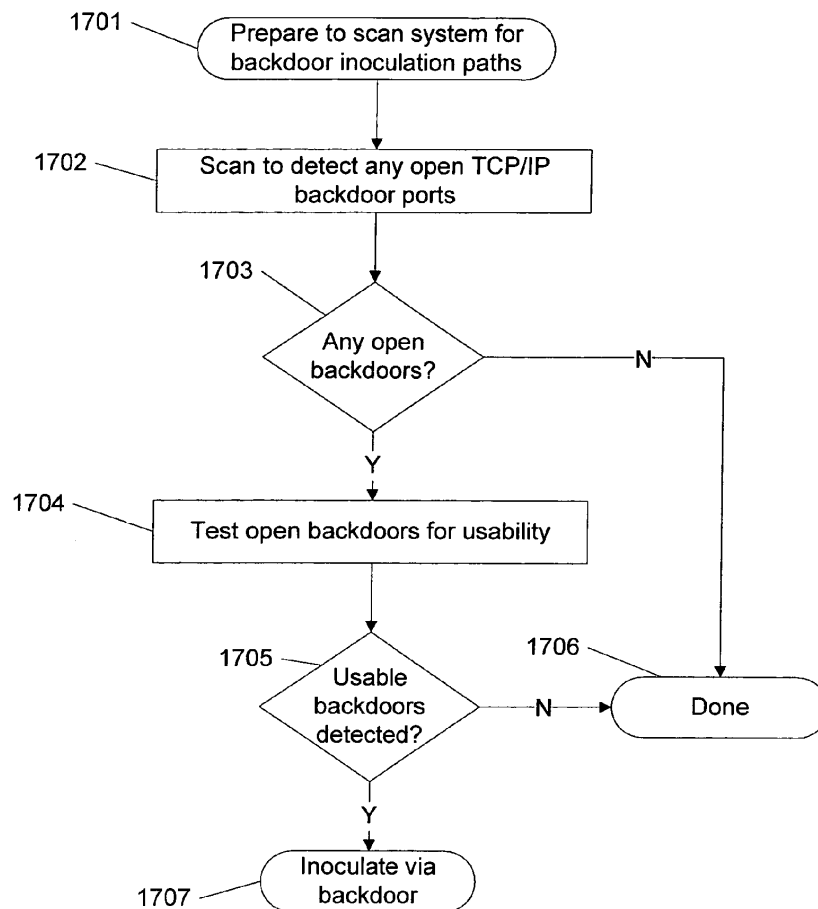


Figure 17

Ex. 1, p. 23

K.Mizra LLC, Ex. 2019, p. 61

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 23 of 25

US 8,234,705 B1

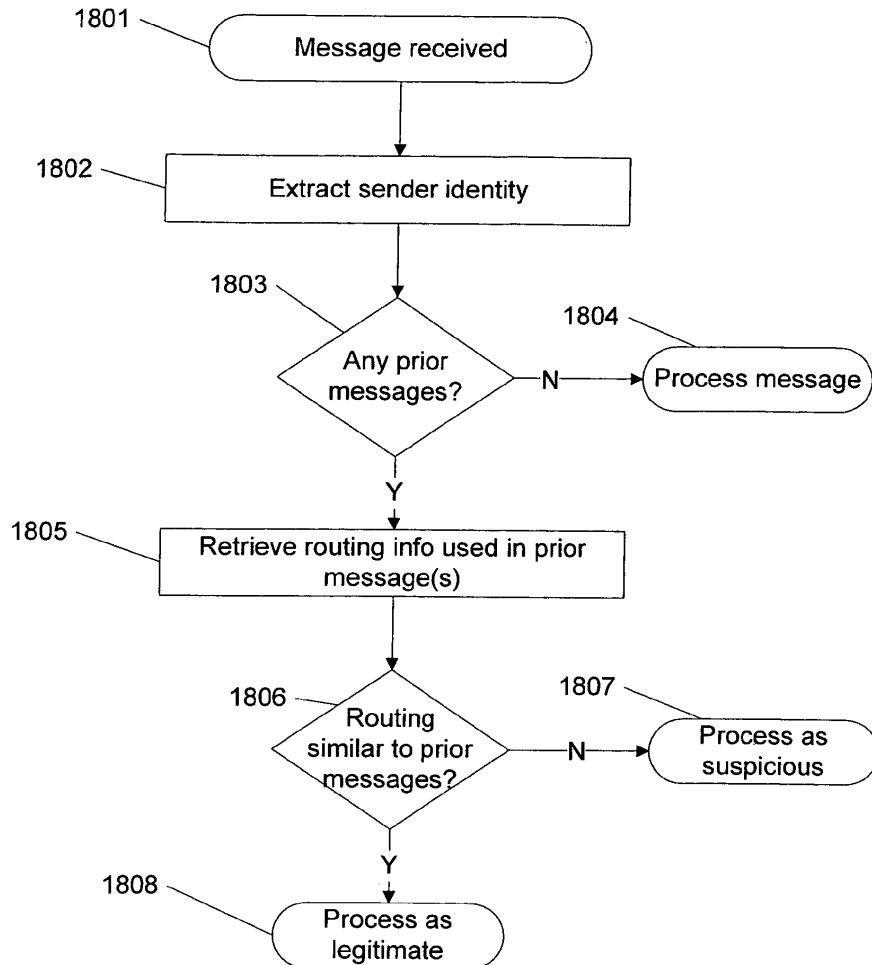


Figure 18

Ex. 1, p. 24

K.Mizra LLC, Ex. 2019, p. 62

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 24 of 25

US 8,234,705 B1

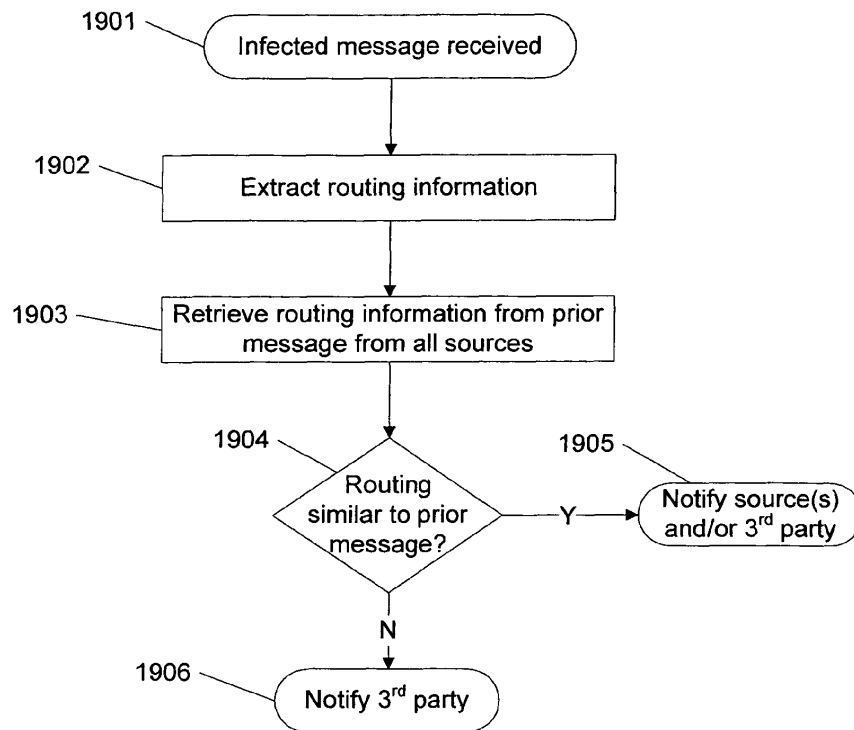


Figure 19A

Ex. 1, p. 25

K.Mizra LLC, Ex. 2019, p. 63

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Jul. 31, 2012

Sheet 25 of 25

US 8,234,705 B1

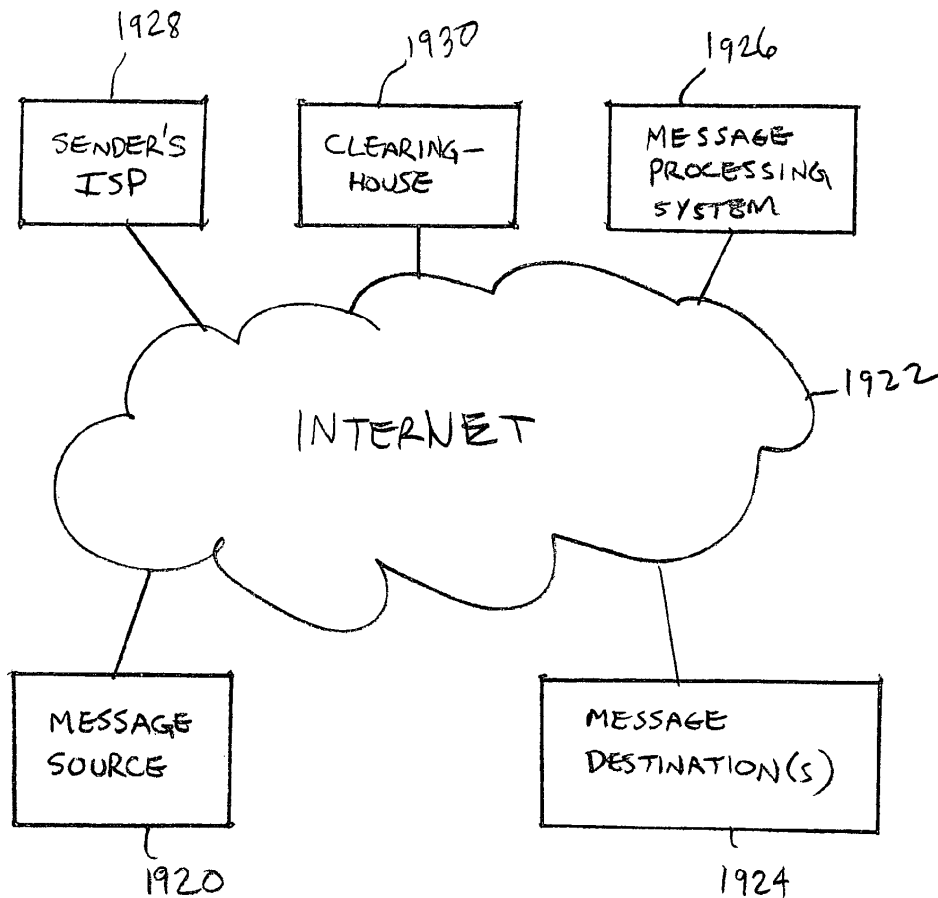


FIG. 19B

Ex. 1, p. 26

K.Mizra LLC, Ex. 2019, p. 64

Google LLC v. K.Mizra LLC

IPR2025-01436

US 8,234,705 B1

1

**CONTAGION ISOLATION AND
INOCULATION****CROSS REFERENCE TO OTHER
APPLICATIONS**

This application claims priority to U.S. Provisional Patent Application No. 60/613,909 entitled CONTAGION ISOLATION AND INOCULATION filed Sep. 27, 2004, which is incorporated herein by reference for all purposes.

BACKGROUND OF THE INVENTION

Laptop and wireless computers and other mobile systems pose a threat to elements comprising and/or connected to a network service provider, enterprise, or other protected networks to which they reconnect after a period of connection to one or more networks and/or systems that are not part of the service provider, enterprise, or other protected network. By roaming to unknown domains, such as the Internet, and/or connecting to such domains through public, wireless, and/or otherwise less secure access nodes, such mobile systems may become infected by computer viruses, worms, backdoors, and/or countless other threats and/or exploits and/or have unauthorized software installed; have software installed on the mobile system by an operator of the protected network for the protection of the mobile system and/or the protected network removed or altered without authorization; and/or have configurations, settings, security data, and/or other data added, removed, and/or changed in unauthorized ways and/or by unauthorized person. Similarly, a stationary system may become infected, e.g., due to receipt and execution of malicious code via a network or other communication and/or a diskette and/or other removable media. Upon connecting to a protected network, a system may infect or otherwise harm resources associated with the protected network before measures can be taken to detect and prevent the spread of such infections or harm. Therefore, there is a need for a reliable way to ensure that a system does not infect or otherwise harm other network resources when connected to a protected network.

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

FIG. 2B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments.

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

FIGS. 3B-3D illustrate distributed update distribution as implemented according to some embodiments.

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

FIG. 4B is a block diagram of a client system configured to obtain an update according to some embodiments.

FIG. 4C is a block diagram of a system used to distribute an update according to some embodiments.

2

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

FIG. 10B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined 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 an infestation according to some embodiments.

FIG. 13 is a flow diagram of a method for monitoring one or more computers for infestation 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. 19A is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FIG. 19B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected.

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. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time and a specific component that is manufactured to perform the task. 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 limited only by the claims 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 under-

Ex. 1, p. 27

K.Mizra LLC, Ex. 2019, p. 65
Google LLC v. K.Mizra LLC
IPR2025-01436

US 8,234,705 B1

3

standing of the invention. These details are provided for the purpose of example and the invention may be practiced according to the claims 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 isolation and inoculation is disclosed. In some embodiments, a request is received from a host, e.g., via a network interface, to connect to a protected network. It is determined whether the host is required to be quarantined. If the host is required to be quarantined, the host is provided only limited access to the protected network. In some embodiments, a quarantined host is permitted to access the protected network only as required to remedy a condition that caused the quarantine to be imposed, such as to download a software patch, update, or definition; install, remove, and/or configure software and/or settings as required by a policy; and/or to have a scan or other diagnostic and/or remedial operation performed. In various embodiments attempts to communicate with hosts not involved in remediation are redirected to a quarantine system, such as a server, that provides information, notices, updates, and/or instructions to the user.

Contagion refers herein to software programs and/or code, data values, configurations, and settings, that expose a computer or other system to security threats, such as unauthorized access and/or use of the system and/or data stored thereon; corruption or loss of data; and/or damage, loss, and/or unauthorized reconfiguration of system software and/or hardware components. Examples of contagion include software programs and/or code designed to perform undesirable and/or unauthorized 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 refers herein to a 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 detected to be malicious. Recent activity may be evaluated (102) to see if it is anomalous. Examples of anomalous activity include median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, and 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

4

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 anomalous (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. Another example of reducing the next update interval is to reduce the currently remaining time until an update, for example reducing by a proportion, such as 50%, the remaining time until an update is requested.

FIG. 2A 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. 3A, which may include updating or concluding that an update is not needed. A time for next contacting an anti-contagion source is decided upon (204). In some embodiments, the scheduled time may be made sooner when the source of the update, such as a vendor of software to which the update pertains, 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. 2B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments. A plurality of clients 1 to "n", represented in FIG. 2B by clients 220, 222, and 224, are connected to the Internet 226. An update server 228 also is connected to the Internet 226. In various embodiments, one or more of the clients 220-224 is configured to request and obtain updates, if any, from update server 228 via the Internet, e.g., as described above in connection with FIG. 2A.

FIG. 3A 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 (301), for example as was discussed in 202 of FIG. 2A. 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 pref-

Ex. 1, p. 28

US 8,234,705 B1

5

erence 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. 4A. After the update has been acquired (305), the contact may be complete (306).

Distributed update distribution is disclosed. In some embodiments, at least some users receive an update indirectly, e.g., from another user that received it previously, either directly from the original source or indirectly from a source that had itself earlier received it, again either directly or indirectly, and so on. FIGS. 3B-3D illustrate distributed update distribution as implemented according to some embodiments. In the example shown in FIGS. 3B-3D, it is assumed for expository purposes that each host that receives the update will serve as a redistribution node, each recipient has the ability to redistribute at the same rate as the original source, and there is continuous demand for the update from all sources. As shown in FIG. 3B, at a time t_1 a first direct recipient of the update D1 (332) downloads the update from the original source S (330). As shown in FIG. 3C, at a time t_2 a second direct recipient D2 (334) downloads the update from the original source and an indirect recipient R11 (336) downloads the update from the first direct recipient D1 (332), which in this case acts as a redistribution node. As shown in FIG. 3D, at a time t_3 a third direct recipient D3 (338) downloads the update from original source S (330), a second indirect recipient R12 (340) downloads the update from the first direct recipient D1 (332), a third indirect recipient R21 (342) downloads the update from the second direct recipient D2, and a fourth indirect recipient R111 (344) downloads the update from first indirect recipient R11 (336). In this example, in the three download cycles shown in FIGS. 3B-3D on its own the original source would only have been able to download the update to the three direct recipients, but by using previous recipients to redistribute, in the example shown in FIGS. 3B-3D a total of seven recipients have been able to obtain the update. In a generalized case, under the assumptions used in this example in "n" download cycles/periods a total of $2n-1$ recipients are able to obtain the update, whereas only "n" recipients, i.e., one per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source. In an even more generalized case, under an assumption that each server may download roughly k copies in a time interval, for example where more than one download is provided by a server at a time, then in "n" download cycles (or periods) a total of roughly $(k+1)n-1$ recipients are able to obtain the update, where as only $k*n$ recipients, i.e., k per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source.

FIG. 4A is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an

6

update source is contacted (401), such as via an Internet connection, for example via TCP/IP. An offer to serve as a redistribution source is made (402). In some embodiments, 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. In various embodiments, 402 is optional, performed only if a criterion or test is satisfied, or omitted. An update or an update identifier is 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 confirms redistribution sourcing status (408) in this example. In some embodiments, confirmation includes 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 acquired (405) in this example. In some embodiments, the list is or may be explicit. In some embodiments, the list is or 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 are dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources 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. In some embodiments, a requesting host that agrees to serve as a redistribution source is provided with a list of sources from which the update is available immediately, or at least sooner than it will be made available to hosts that will not redistribute. One or more 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 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 or within a recipient's intranet or home network. If a reservation time was acquired, then a wait for that reservation time (410) is performed in this example. At the reservation time, a further attempt to acquire an update or update identifier from the source (403) is made, and the flow proceeds as described earlier. This flow may continue until an acquisition is complete (408), as also described earlier. In various embodiments, one or more of confirming redistribution sourcing status (408), attempting to obtain the update from a redistribution source (405-407), and/or acquiring and/or waiting for a reservation time (409-410) is/are optional, conditional, or omitted.

FIG. 4B is a block diagram of a client system configured to obtain an update according to some embodiments. In some embodiments, the client system shown in FIG. 4B is used to implement one or more of clients 220-224 of FIG. 2B. The client system 420 includes a communication interface 422,

Ex. 1, p. 29

US 8,234,705 B1

7

such as a network interface card, configured to send/receive communications via a network, such as the Internet. The client system 420 further includes an update agent and/or browser 424. In some embodiments, the update agent/browser 424 comprises browser software used to navigate the World Wide Web, e.g., to send/receive data to/from a website from which an update is available for download. In some embodiments, update agent/browser 424 includes an agent associated with anti-contagion software installed on the client system 420. In various embodiments, the update agent/browser 424 comprises one or more components and is implemented as one or more software programs running on a processor and/or one or more hardware components. A settings data store 426 stores one or more settings and/or other data associated with the software with which an update and/or software to which the update does or may pertain, such as periodicity or next scheduled (e.g., reservation) time to check for updates, a list of update identifiers, locators, and/or redistribution sources from which an update is or may be available, policies for whether and/or under what terms, conditions, circumstances, etc. the client system 420 may be used redistribute an update, a last update time, an old version identifier (see, e.g., 202 of FIG. 2A), etc.

FIG. 4C is a block diagram of a system used to distribute an update according to some embodiments. In some embodiments, the system of FIG. 4C is used to provide an update server, such as update server 228 of FIG. 2B. In this example, the distribution system 440 includes a communication interface 442 configured to send/receive data via a network, such as the Internet. An update distribution engine 444 is configured to receive and service requests for updates. For updates provided directly to a request source, the update distribution engine 444 obtains the update from an update storage location 446 and provides the update to the request source via the communication interface 442. In some embodiments, if data included in the request indicates the source of the request is or may be willing to serve as a redistribution source, the update distribution engine 444 hands the request off to a redistribution coordinator 448, which negotiates with the request source, e.g., based on policies, settings, rules, etc. stored in a policies and lists data store 450, through communications sent/received via communication interface 442, to determine if the requesting source will serve as a redistribution source and, if so, to determine the terms and conditions of such service and data, such as an address, locator, or identifier, that can be provided to other requesting systems and/or users to enable them to attempt to obtain the update from the redistribution source. In some embodiments, if update distribution engine 444 determines that it is not able to provide the update to the request source directly (e.g., due to capacity limits or because a policy applicable to the request and/or the request source indicates the request source should be directed to a redistribution source, e.g., because the request source has a less expensive version of the software or is not willing to serve as a redistribution source), the update distribution engine 444 hands the request off to redistribution coordinator 448, which provides to the request source an update identifier and a list of one or more redistribution sources from which the update is or may be available to the request source.

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

8

update is multicast (504) in this example. In some embodiments, a multicast 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 a provided 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 a 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/or 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 pro-

Ex. 1, p. 30

US 8,234,705 B1

9

cessed, 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 10 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 15 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 20 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 25 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 a user cannot modify.

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 a request 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 demonstrates that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source allows 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. An authentication to be used to authenticate requests is negotiated (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 30

10

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. 4A. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect 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 or authorized 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. A next address and (optional) authenticator in the list of sources is retrieved (802). Any order may be used. In some embodiments, an order of processing may correspond to the original list order. In some 35 embodiments, an order of processing may be such that the earlier processed addresses would be expected to have lower latency and/or higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list are or 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), which may in some embodiments be performed as described in conjunction with 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 a connection such as a TCP/IP connection with the source. If the acquisition is complete (804), then the processing is complete (805) in this example. If the acquisition 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. Examples of a server include an HTTP server, an FTP server, and a multicast server. A server waits 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 receiving and validating, or perform-

Ex. 1, p. 31

US 8,234,705 B1

11

ing a mutual proof of possession of, 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 considered 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/or an appropriate coding, such as uuencode.

FIG. 10A 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 presence and/or 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 FIGS. 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 the problem is quarantined (1002). In some embodiments, quarantining includes preventing 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 are authorized. In some embodiments, quarantining includes preventing undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIGS. 14 and 15. In this example, the quarantined computer is provided access to remediation services (1003). In some embodiments, access to remediation includes or 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 includes allowing 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 is restricted by the quarantine to 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. 10B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments. A plurality of hosts 1 to "m", represented in FIG. 10B by hosts 1020, 1022, and 1024, connect via a router 1026 and a gateway 1028 to the Internet 1030. In various embodiments the router 1026 is omitted or included in the gateway 1028. In various embodiments, gateway 1028 comprises a gateway, router, firewall, or other device configured to provide and control access between a protected network and the Internet and/or another public or private network. In various embodiments, the pro-

12

cess of FIG. 10A is implemented, in the event of quarantine of one or more of hosts 1020-1024, for example, on the affected (quarantined) host, router 1026, and/or gateway 1028. In some embodiments, a quarantined host (or a host associated with a quarantined network or sub-network) is permitted to access a remediation server 1032, e.g., to download a patch, more current threat definition, etc. In some embodiments, requests to connection to a host other than the remediation server 1032 are redirected to a quarantine server 1034 configured to provide a notice and/or other information and/or instructions to a user of the quarantined host. In various alternative embodiments the quarantine server 1034 is omitted or resides on the router 1026 and/or gateway 1028.

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 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 scanning 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 no open ports are detected (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 is performed or attempted (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 consequent errant response, where the errant response may for example include a characteristic communication, 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 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

Ex. 1, p. 32

US 8,234,705 B1

13

example by checking in DHCP or static IP allocation tables for the subscriber's identity, or for example by extracting a MAC address from traffic associated with the IP address, 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 an infestation according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, is used in some embodiments to monitor a computer for infestation (1201). Network traffic is monitored for anomalous traffic volume, port, or IP destination (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 a 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 inoculation, 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 one or more computers for infestation according to some embodiments. In this example monitoring of a 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. A computer associated with an address identified in a list is queried for a cleanliness assertion (1302), for example by contacting a trusted computing base within a computer, and requesting an

14

authenticated infestation scan by trusted software. An example of a trusted computing base within a computer is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Another example of a trusted computing base is described in various TCG specifications, such as the TCG Architecture Overview, published by the Trusted Computing Group. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and/or operating system. In some embodiments, trusted code bases may digitally sign assertions about the cleanliness (e.g. infestation status) and/or 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 a version associated with a current anti-contagion software or definition file in use, wherein a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with information associated with its patch level, wherein a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (1303) then monitoring is 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 is 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 is tested in this example 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, such as an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (1403), then the traffic is 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

Ex. 1, p. 33

US 8,234,705 B1

15

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 request is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine server that provides remediation information. If the request was not for an approved service (1406), then the outbound traffic is discarded or responded to with an 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 of FIG. 14. Inbound traffic originally destined for a quarantined computer is 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 is tested in this example 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. In various embodiments, at least certain types of traffic from approved sites such as anti-contagion vendors, security update providers, and/or internal ISP sites are 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 an 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, a device such as a router may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in conjunction with 1407 of FIG. 14. If a received connection is a web server request (1603), such as an HTTP

16

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, and/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 is 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 partnered 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. A short time to live ensures that a further DNS query will be sent prior to subsequent communications being sent from the quarantined computer to the host that was the subject of the DNS query, aiding a return to normal routing of traffic once the computer is no longer under quarantine. 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 information associated with 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. In some embodiments, a user is requested and/or required to authorize in advance use of an authorized and/or any unauthorized backdoor for purposes of inoculation. 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 weak-

Ex. 1, p. 34

US 8,234,705 B1

17

ness 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 one or more backdoor ports are open (1703) then they are tested for usability (1704) in this example. In some embodiments, a backdoor port may be assumed open (1703) and 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 backdoor 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 an optional 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 is extracted (1802), for example by examining the “From:” header line in an email message, or by examining the envelope information conveying an email message, or by examining the sender’s ID in an instant message. A check is 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

18

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 is 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 information for a message purporting to be from a sender is not similar to previous messages from the same sender (1806) then the message is 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. In various embodiments, the process of FIG. 18 is implemented on a mail, messaging, or other server associated with the sender and/or the sender’s ISP; a mail, messaging, or other server associated with the recipient and/or the recipient’s ISP; and/or one or more intermediate nodes.

FIG. 19A 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 as such by anti-contagion software. Routing information is 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 is 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 is compared with the retrieved routing information (1904). If no similarity with previous messages is found (1904) then in this example a third party notification is made (1906). An example of a third party notification is sending notification to the ISP

Ex. 1, p. 35

US 8,234,705 B1

19

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 service that determines the credibility of a report, such as a collaborative filter service, that accumulates accusations of viral disseminations. 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 infected message source. If a similarity is found with previous messages (1904) then in this example a notification is sent (1905). 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.

FIG. 19B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected. A message source 1920 sends a message via the Internet 1922 to a message processing system 1926 for delivery via the Internet to one or more message destinations 1924. In various embodiments, the message processing system 1926 comprises a mail or messaging server associated with the message source 1920, a mail or messaging server associated with the destination(s) 1924, and/or an intermediate processing node. In some embodiments, the message source 1920 is connected directly and/or via a local network, and not via the Internet 1922 as shown in FIG. 19B, to message processing system 1926. In some embodiments, message processing system 1926 is configured to identify suspicious messages using current and historic routing information, e.g., as described above in connection with FIG. 18. In some embodiments, the process of FIG. 19A is implemented on message processing system 1926. For example, in some embodiments if message processing system 1926 determines that a message sent by message source 1920 is infected, message processing system 1926 compares routing information associated with the malicious message with corresponding information for one or more messages received previously from the message source 1920. If the routing information is similar, in some embodiments the message processing system 1926 notifies the message source 1920. If the routing information is not similar, in various embodiments the message processing system 1926 notifies the message source 1920, the sender's ISP 1928, and/or a clearinghouse or similar applicable service 1930, as described above.

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.

What is claimed is:

1. A method for protecting a network, comprising: detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the

20

first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and permitting the first host to communicate with the remediation host.

2. A method as recited in claim 1, wherein detecting an insecure condition further includes at least one of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

3. A method as recited in claim 1, wherein detecting an insecure condition includes determining that the first host should be quarantined until an update to an operating system has been installed.

4. A method as recited in claim 1, wherein detecting an insecure condition includes configuring an operating system to quarantine the first host upon initial startup after installation of the operating system.

5. A method as recited in claim 1, wherein permitting the first host to communicate with the remediation host includes: detecting an outbound communication from the first host; and

forwarding the outbound communication if it is addressed to the remediation host.

6. A method as recited in claim 1, wherein preventing the first host from sending data to the one or more other hosts includes:

detecting an outbound communication from the first host; and

redirecting the outbound communication to a quarantine server if it comprises a request for an approved service and is not addressed to a remediation host.

7. A method as recited in claim 1, wherein quarantining the first host further includes preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

8. A method as recited in claim 1, performed at an Internet service provider.

9. A method as recited in claim 1, wherein the software component on the first host is an operating system.

10. A method as recited in claim 1, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

11. A method as recited in claim 1, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

Ex. 1, p. 36

US 8,234,705 B1

21

12. A system for protecting a network, comprising:
a processor configured to:

detect an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantine the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permit the first host to communicate with the remediation host; and
a memory coupled to the processor and configured to provide instructions to the processor.

13. A system as recited in claim 12, wherein the processor is configured to detect an insecure condition at least in part by performing one or more of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

14. A system as recited in claim 12, wherein the processor is configured to detect an insecure condition at least in part by determining that an initial startup after installation of an operating system is being performed.

15. A system as recited in claim 12, wherein the processor is configured to quarantine the first host at least in part by

22

preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

16. A system as recited in claim 12, wherein the software component on the first host is an operating system.

17. A system as recited in claim 12, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

18. A system as recited in claim 17, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

19. A computer program product for protecting a network, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, serving a quarantine notification page to the first host when the service request comprises a web server request, and in the event the service request comprises a DNS query, providing in response an IP address of a quarantine server configured to serve the quarantine notification page if a host name that is the subject of the DNS query is not associated with a remediation host configured to provide data usable to remedy the insecure condition; and

permitting the first host to communicate with the remediation host.

* * * * *

EXHIBIT 2



US009516048B1

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

(10) **Patent No.:** **US 9,516,048 B1**

(45) **Date of Patent:** **Dec. 6, 2016**

(54) **CONTAGION ISOLATION AND
INOCULATION VIA QUARANTINE**

(71) Applicants: **Aaron Emigh**, Incline Village, NV
(US); **James Roskind**, Redwood City,
CA (US)

(72) Inventors: **Aaron Emigh**, Incline Village, NV
(US); **James Roskind**, Redwood City,
CA (US)

(73) Assignee: **Radix Holdings, LLC**, Incline Village,
NV (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.: **15/206,227**

(22) Filed: **Jul. 9, 2016**

Related U.S. Application Data

(63) Continuation of application No. 11/237,004, filed on
Sep. 27, 2005, now Pat. No. 9,398,037.

(60) Provisional application No. 60/613,909, filed on Sep.
27, 2004.

(51) **Int. Cl.**
G06F 21/50 (2013.01)
G06F 21/56 (2013.01)
H04L 12/22 (2006.01)
H04L 12/24 (2006.01)
H04L 12/26 (2006.01)
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC **H04L 63/1416** (2013.01); **G06F 21/56**
(2013.01); **H04L 41/082** (2013.01); **H04L**
41/20 (2013.01); **H04L 63/102** (2013.01);
H04L 63/126 (2013.01); **H04L 63/1425**
(2013.01); **H04L 63/1433** (2013.01)

(58) **Field of Classification Search**
CPC .. G06F 21/50–21/577; H04L 41/08–41/0896;
H04L 41/20; H04L 41/28; H04L
63/08–63/205
USPC 726/22–25, 2–15
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,823,479 B1	11/2004	McElhaney et al.
6,996,845 B1	2/2006	Hurst et al.
7,263,720 B2	8/2007	Silvester
7,287,278 B2	10/2007	Liang
7,424,610 B2	9/2008	Ranganathan
2002/0083328 A1	6/2002	Riordan
2003/0188173 A1	10/2003	Zimmer et al.
2003/0188194 A1	10/2003	Currie et al.
2005/0018618 A1	1/2005	Mualem et al.
2005/0033987 A1	2/2005	Yan et al.
2005/0131997 A1	6/2005	Lewis et al.
2006/0053426 A1	3/2006	Dive-Reclus et al.

OTHER PUBLICATIONS

Jing F. Sims, "Office Action, U.S. Appl. No. 11/237,003," U.S. Patent and Trademark Office, Jun. 21, 2012. Relevant passage on pp. 13-15 in the section entitled "Examiner's Statement of Reasons for Allowance."

Mark V. Glover, "Integrating a Trusted Computing Base Extension Server and Secure Session Server into the LINUX Operating System." Sep. 2001, Naval Postgraduate School.

"OLS: Linux and Trusted Computing." Jul. 22, 2005; <http://lwn.net/Articles/144681/> (as retrieved on Jul. 15, 2010).

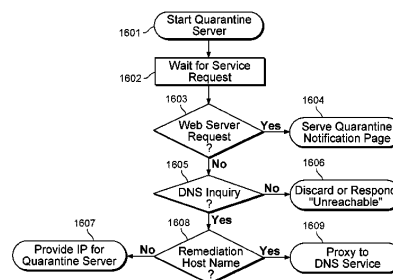
A. Bohra et al., "Remote Repair of Operating System State Using Backdoors." Proceedings of International Conference on Automatic Computing on May 17-18, 2004, pp. 256-263.

Primary Examiner — Timothy J Weidner

(57) **ABSTRACT**

In some embodiments, techniques for protecting a network may include detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, including contacting a trusted computing base and determining whether a response includes a valid digitally signed attestation of cleanliness (an attestation that the first host is not infested and/or that the presence of a patch or patch level of a software component on the first host); if not, quarantining the first host, including by re-routing a service request sent by the first host by responding with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve a quarantine notification page that provides remediation information; and permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

20 Claims, 25 Drawing Sheets



Ex. 2, p. 1

K.Mizra LLC, Ex. 2019, p. 77

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 1 of 25

US 9,516,048 B1

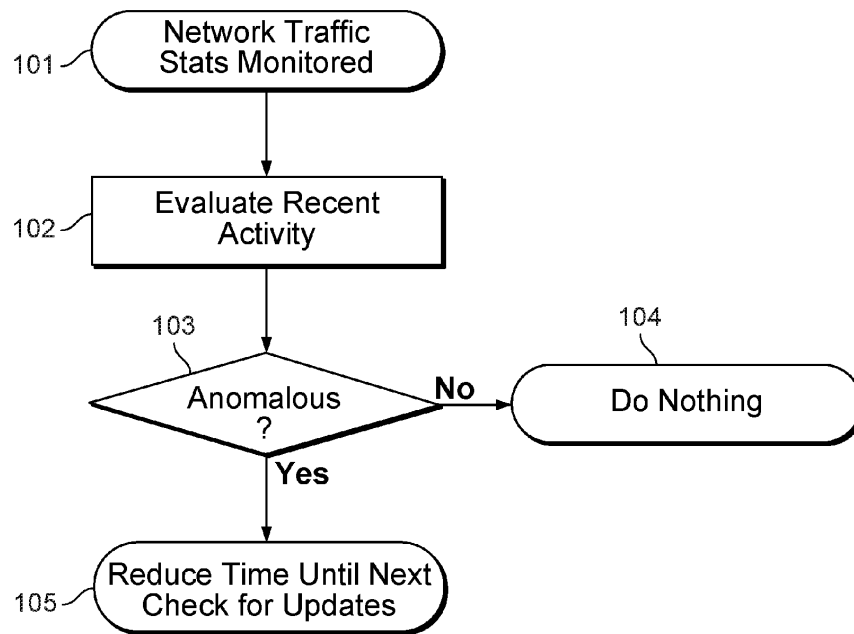


FIG. 1

Ex. 2, p. 2

K.Mizra LLC, Ex. 2019, p. 78

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 2 of 25

US 9,516,048 B1

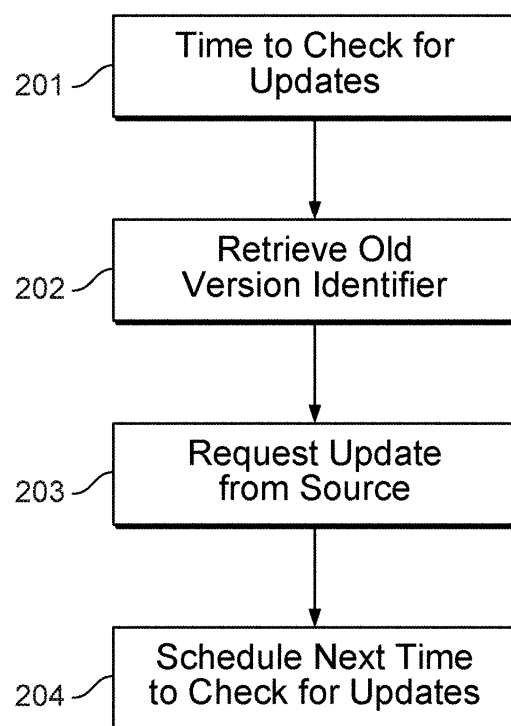


FIG. 2A

Ex. 2, p. 3

K.Mizra LLC, Ex. 2019, p. 79

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 3 of 25

US 9,516,048 B1

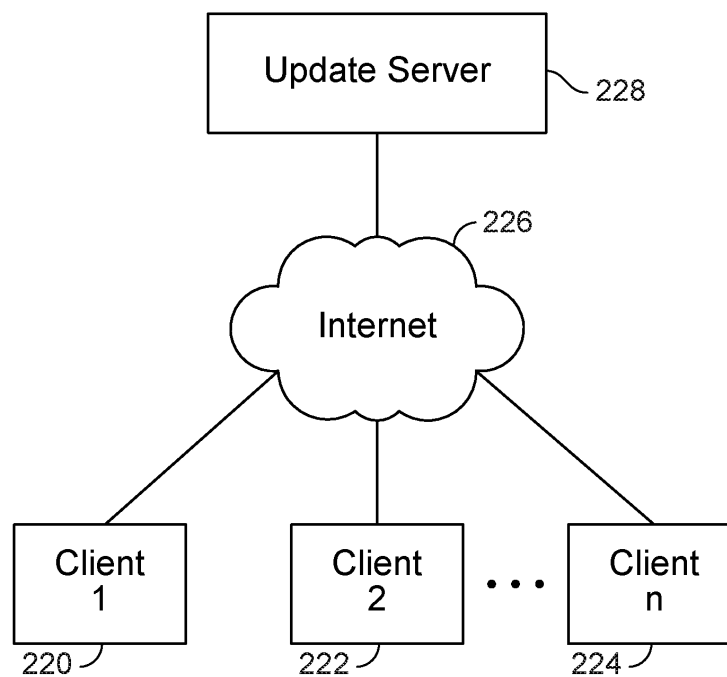


FIG. 2B

Ex. 2, p. 4

K.Mizra LLC, Ex. 2019, p. 80
Google LLC v. K.Mizra LLC
IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 4 of 25

US 9,516,048 B1

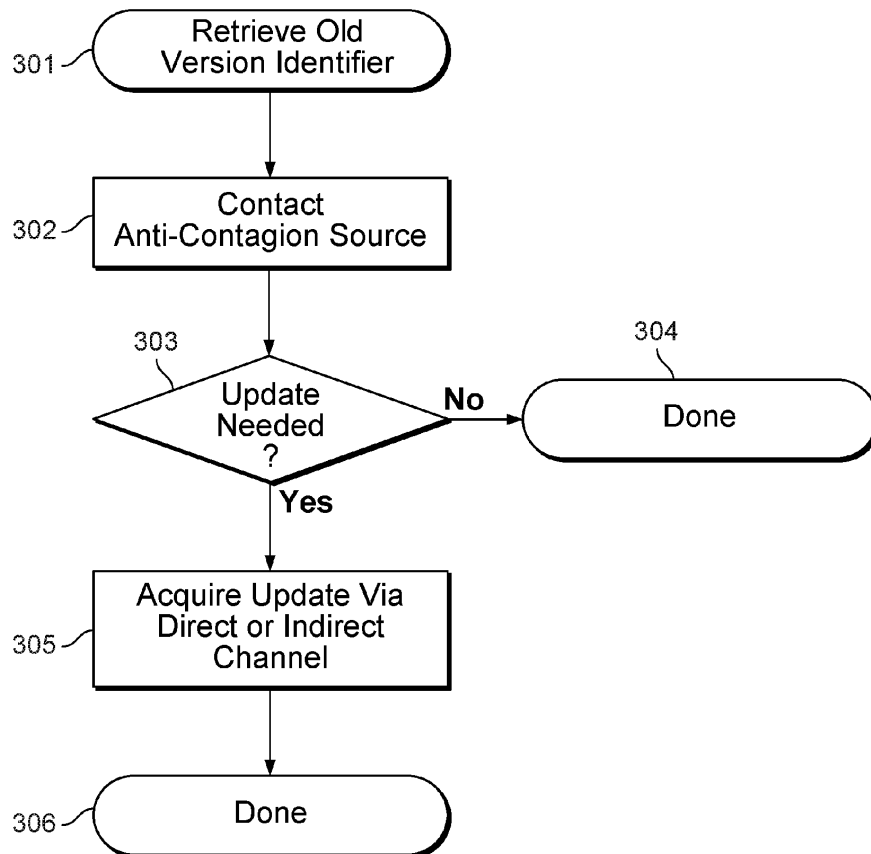


FIG. 3A

Ex. 2, p. 5

K.Mizra LLC, Ex. 2019, p. 81

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 5 of 25

US 9,516,048 B1

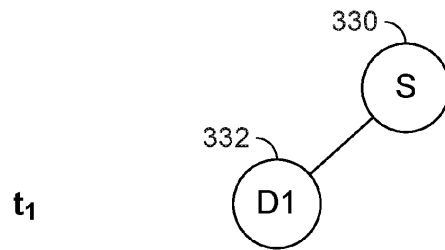


FIG. 3B

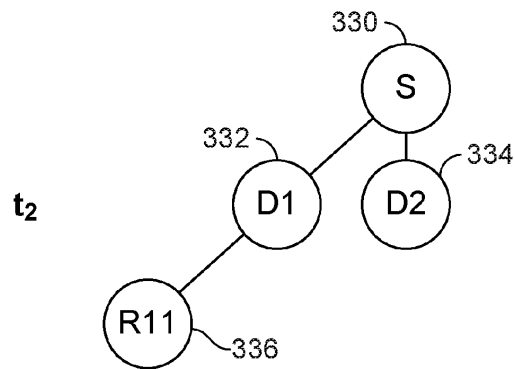


FIG. 3C

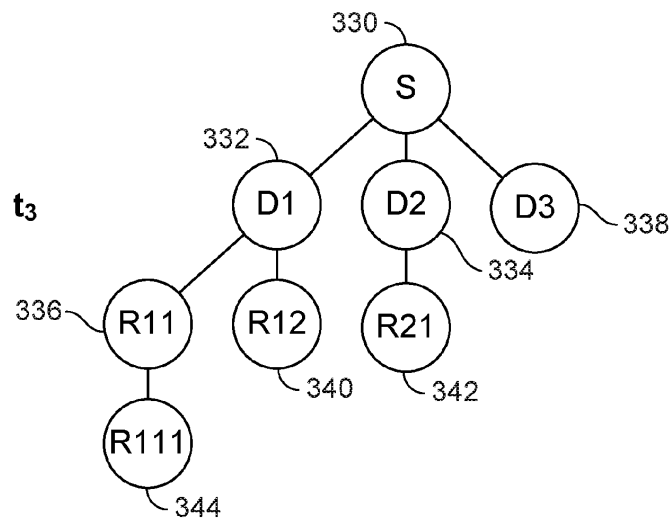


FIG. 3D

Ex. 2, p. 6

K.Mizra LLC, Ex. 2019, p. 82

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 6 of 25

US 9,516,048 B1

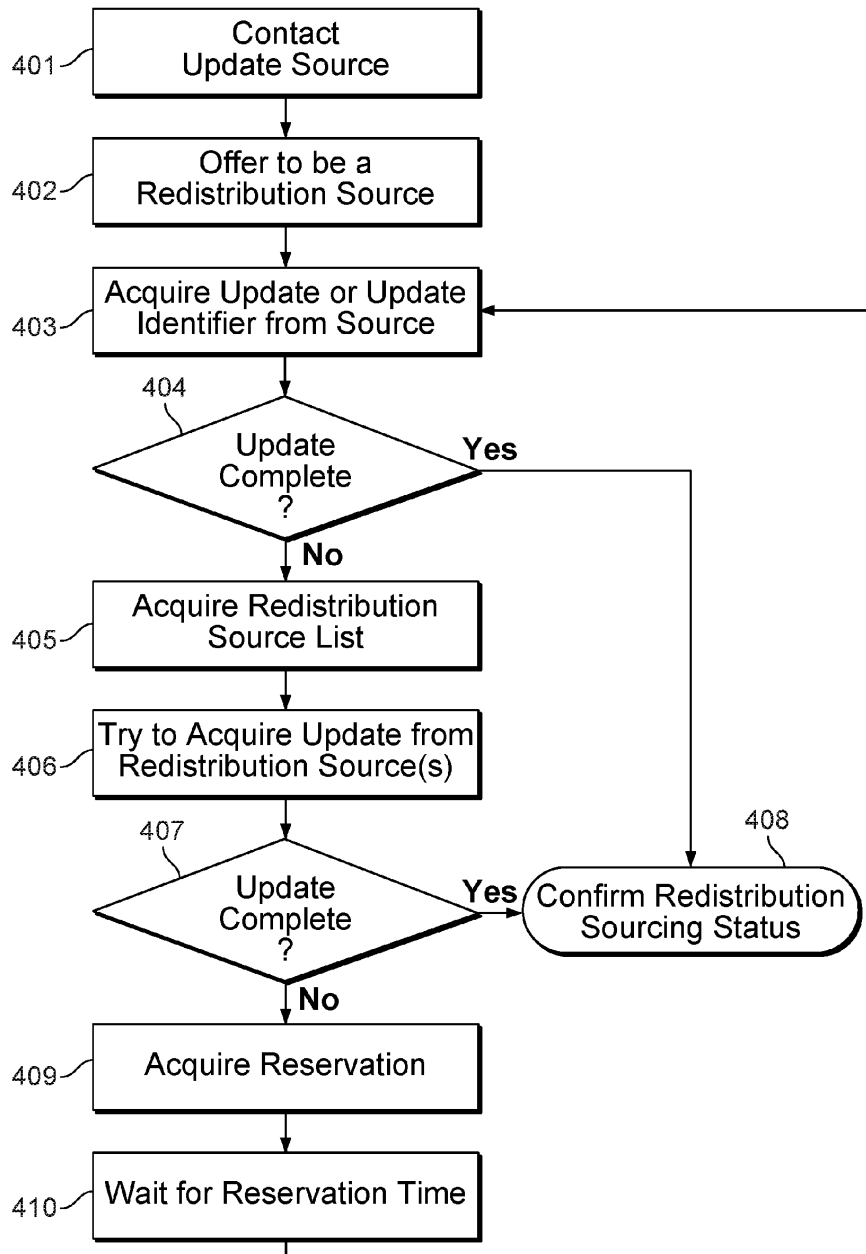


FIG. 4A

Ex. 2, p. 7

K.Mizra LLC, Ex. 2019, p. 83

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 7 of 25

US 9,516,048 B1

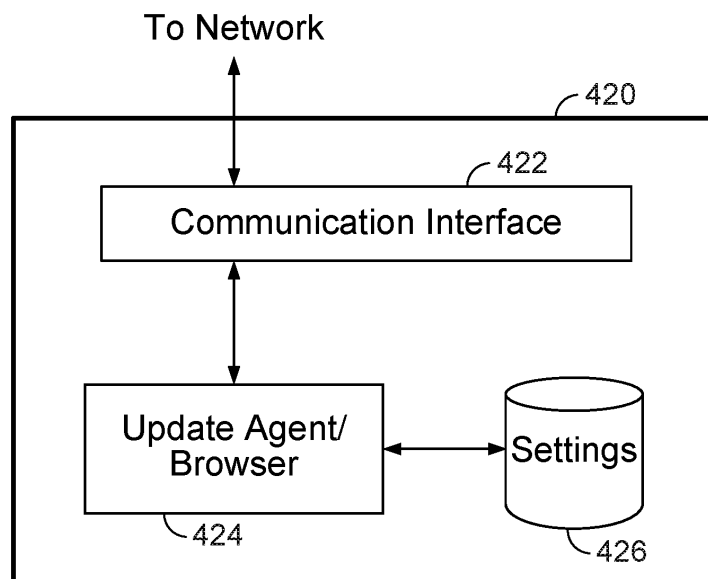


FIG. 4B

Ex. 2, p. 8

K.Mizra LLC, Ex. 2019, p. 84

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 8 of 25

US 9,516,048 B1

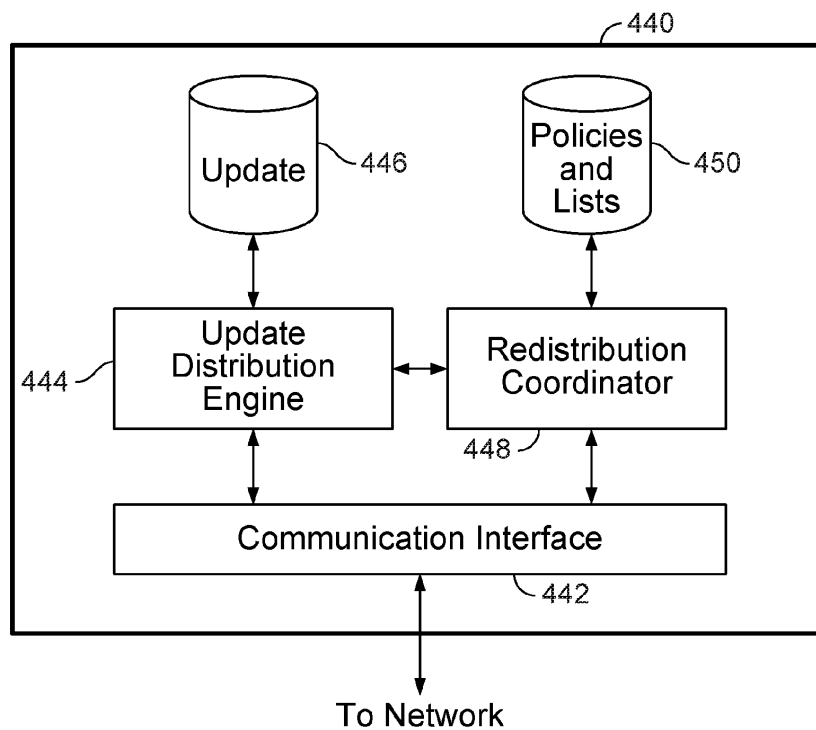


FIG. 4C

Ex. 2, p. 9

K.Mizra LLC, Ex. 2019, p. 85

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 9 of 25

US 9,516,048 B1

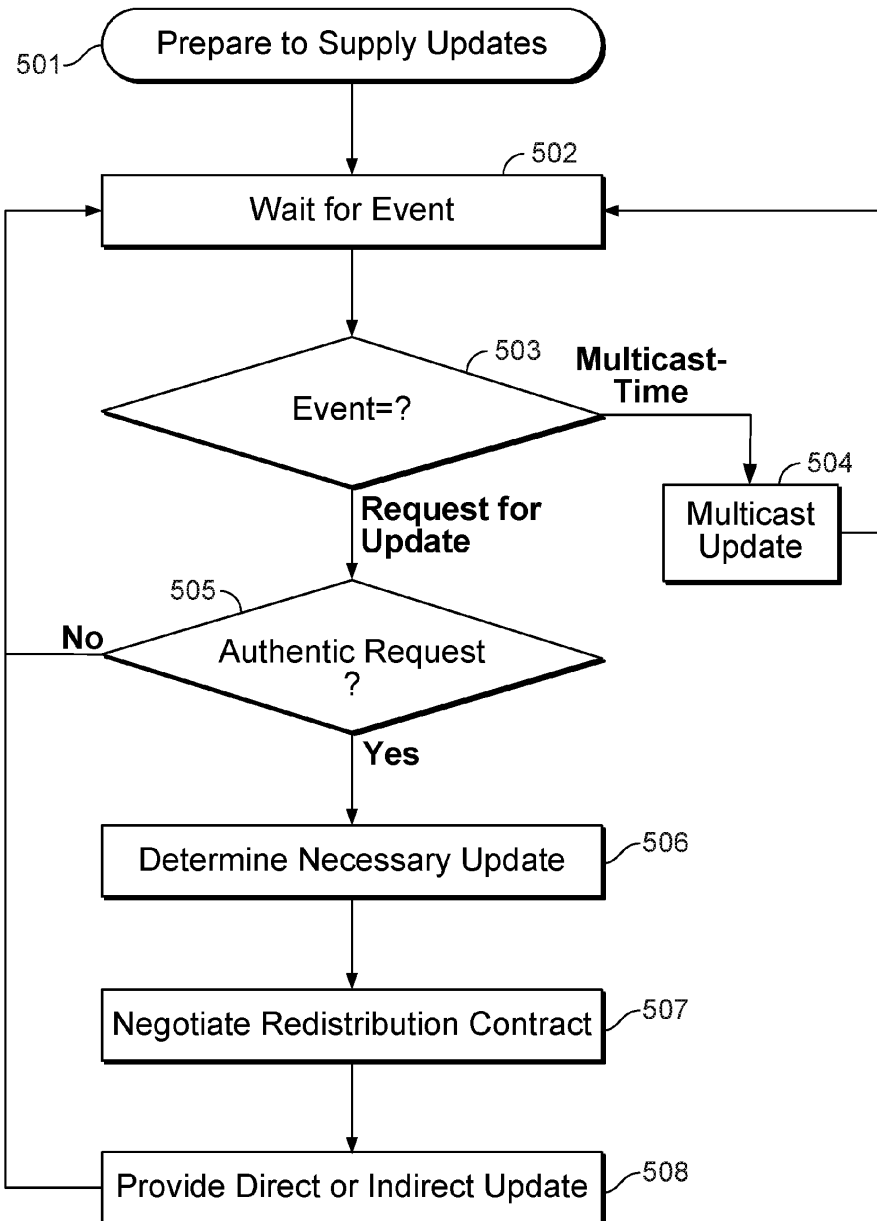


FIG. 5

Ex. 2, p. 10

K.Mizra LLC, Ex. 2019, p. 86

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 10 of 25

US 9,516,048 B1

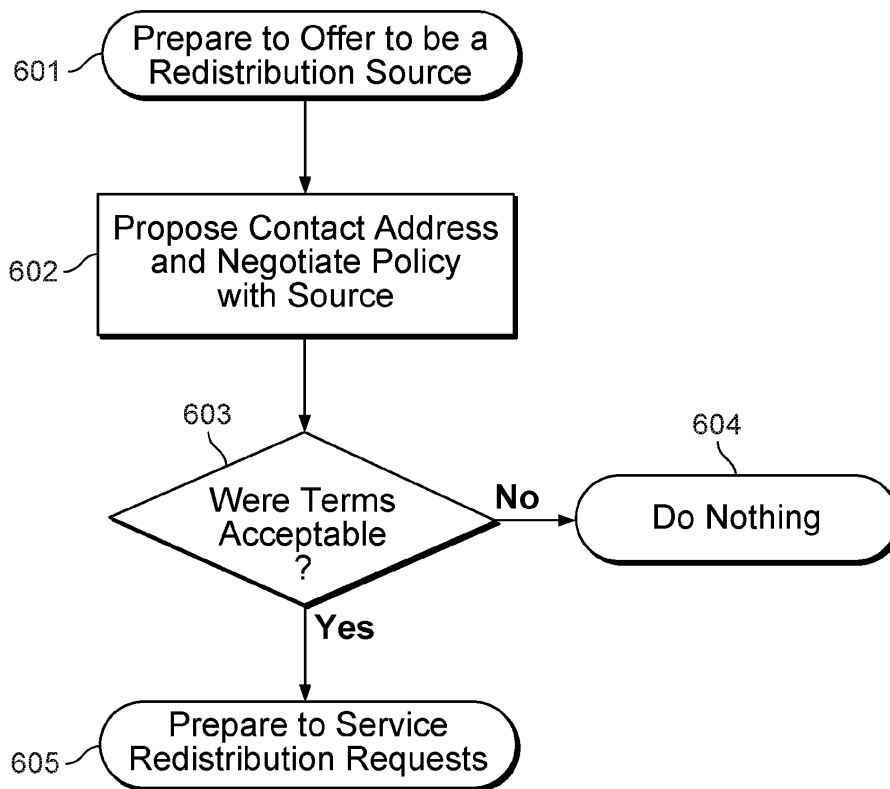


FIG. 6

U.S. Patent

Dec. 6, 2016

Sheet 11 of 25

US 9,516,048 B1

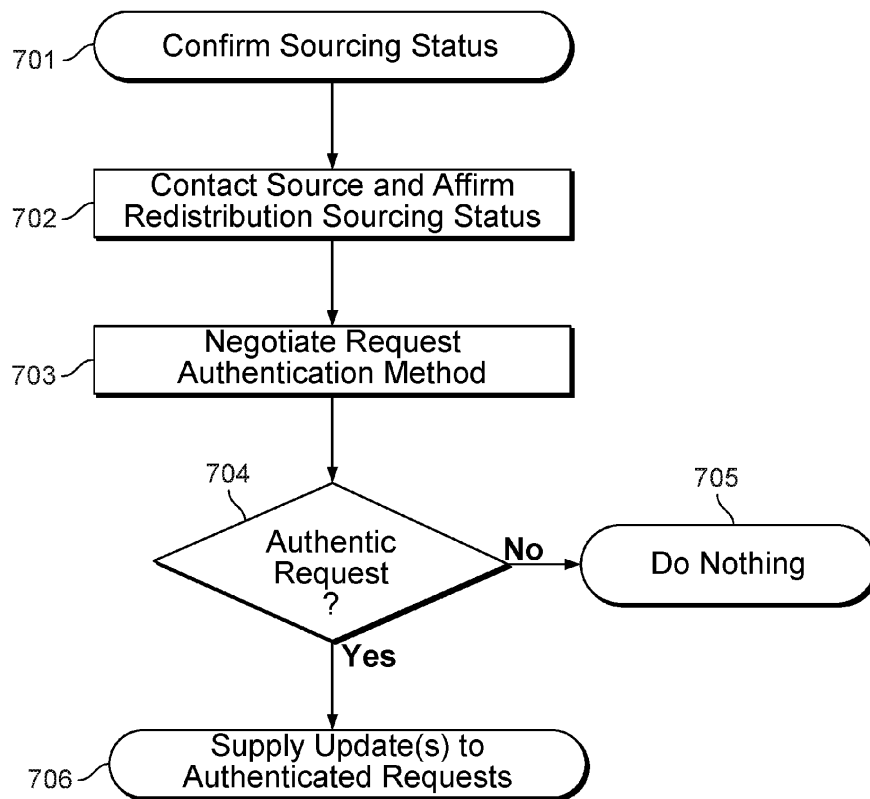


FIG. 7

Ex. 2, p. 12

K.Mizra LLC, Ex. 2019, p. 88

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 12 of 25

US 9,516,048 B1

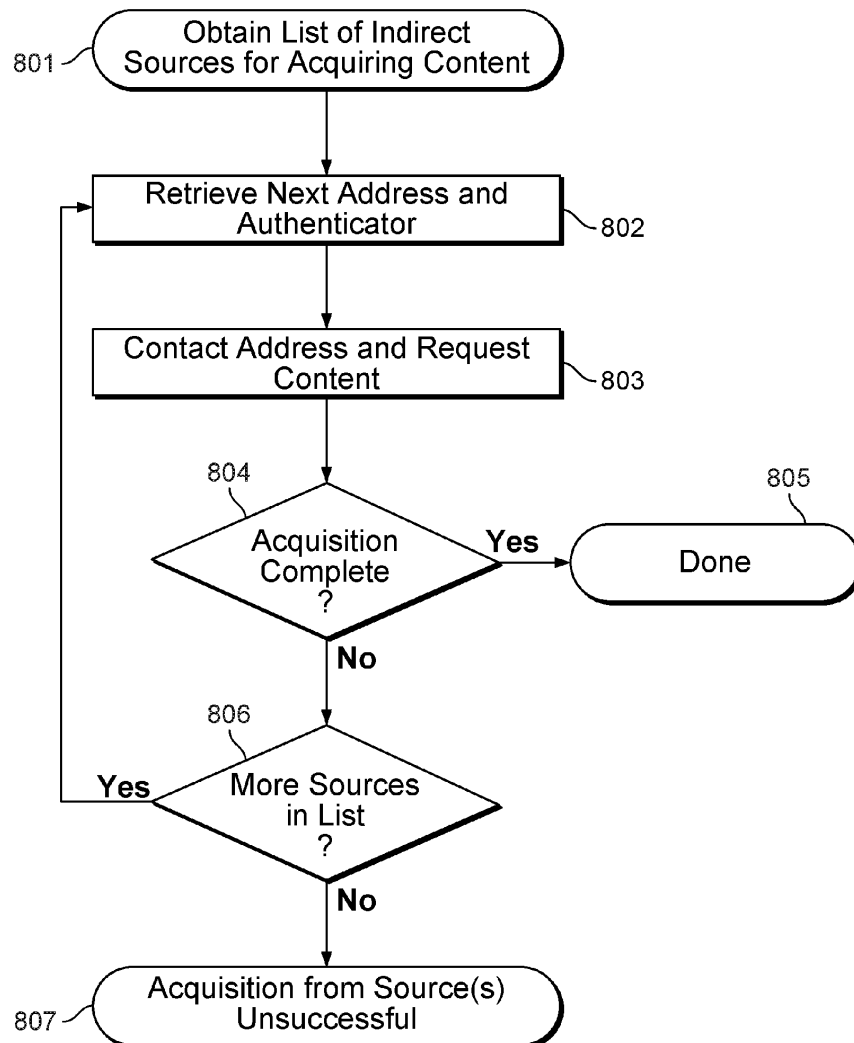


FIG. 8

Ex. 2, p. 13

K.Mizra LLC, Ex. 2019, p. 89

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 13 of 25

US 9,516,048 B1

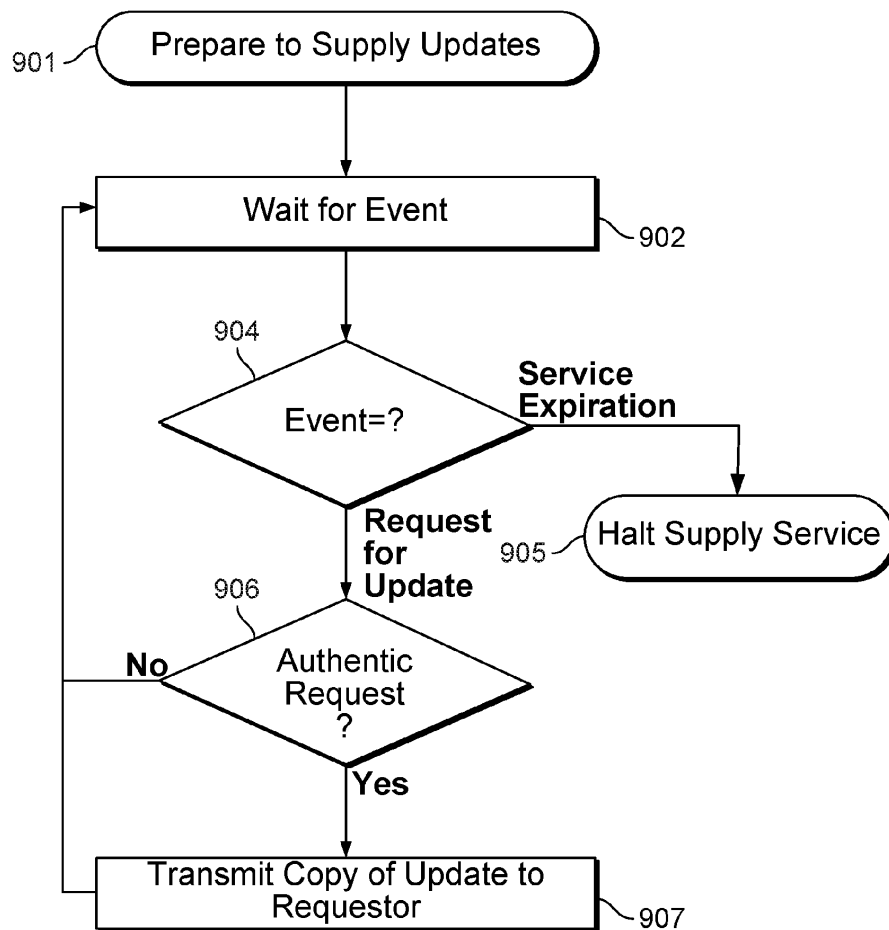


FIG. 9

U.S. Patent

Dec. 6, 2016

Sheet 14 of 25

US 9,516,048 B1

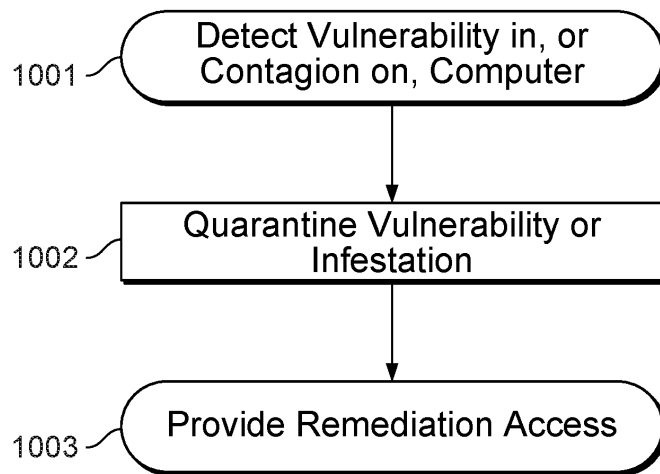


FIG. 10A

Ex. 2, p. 15

K.Mizra LLC, Ex. 2019, p. 91

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 15 of 25

US 9,516,048 B1

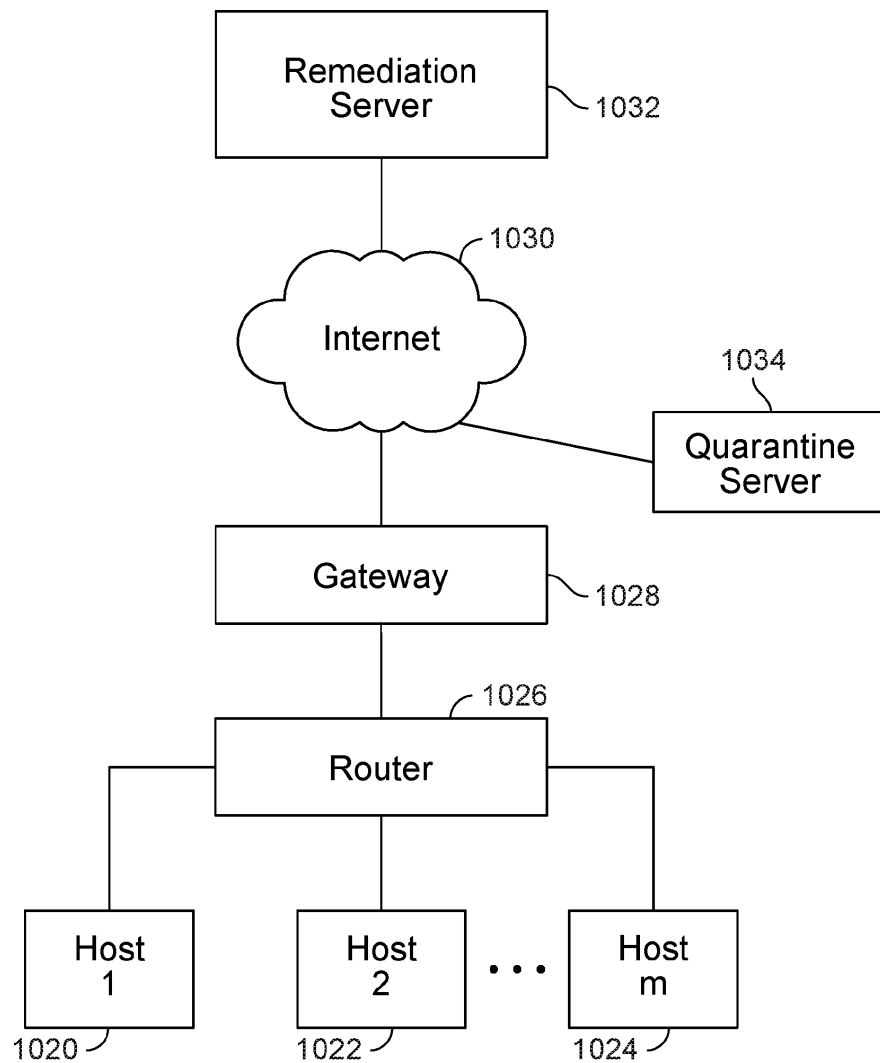


FIG. 10B

Ex. 2, p. 16

K.Mizra LLC, Ex. 2019, p. 92

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 16 of 25

US 9,516,048 B1

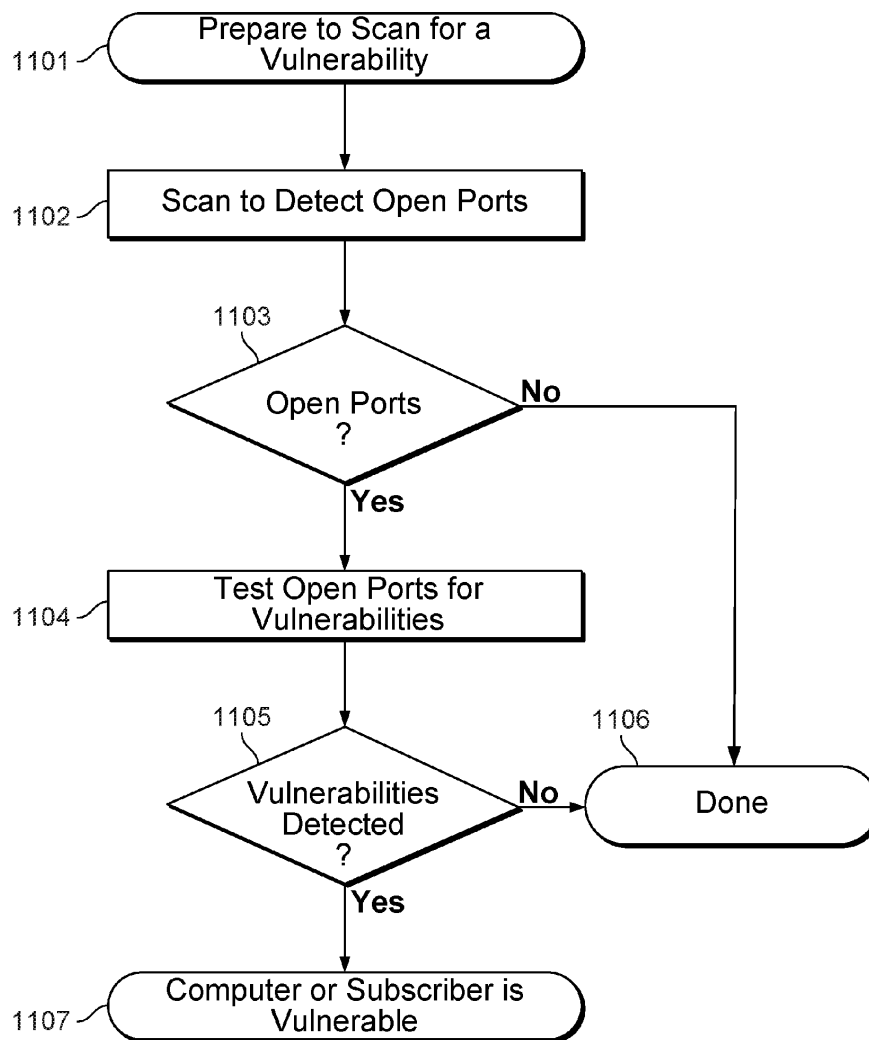


FIG. 11

Ex. 2, p. 17

K.Mizra LLC, Ex. 2019, p. 93

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 17 of 25

US 9,516,048 B1

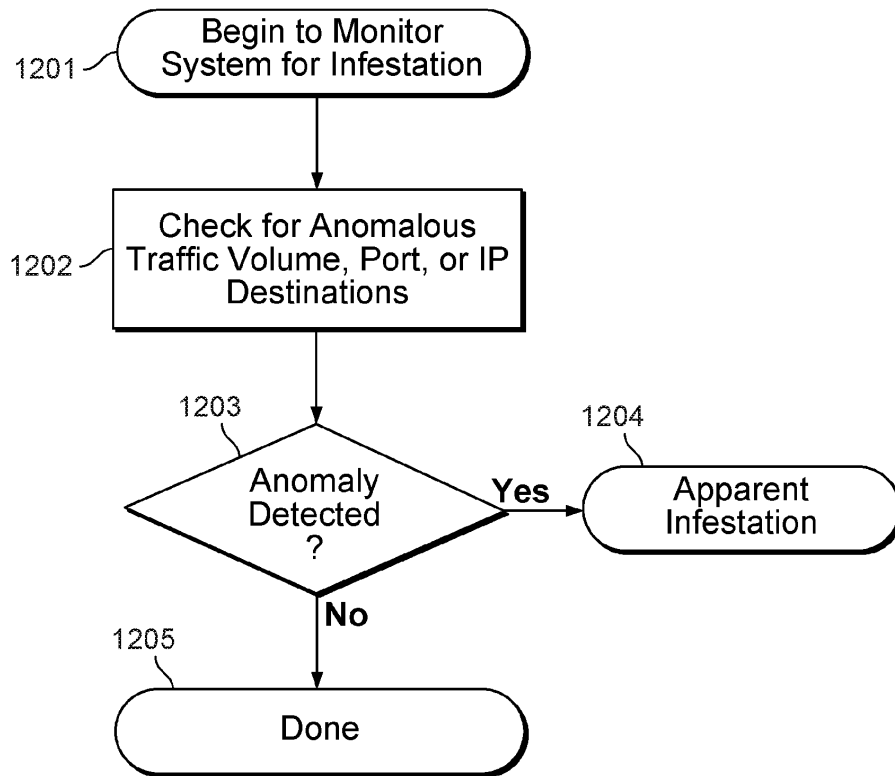


FIG. 12

Ex. 2, p. 18

K.Mizra LLC, Ex. 2019, p. 94

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 18 of 25

US 9,516,048 B1

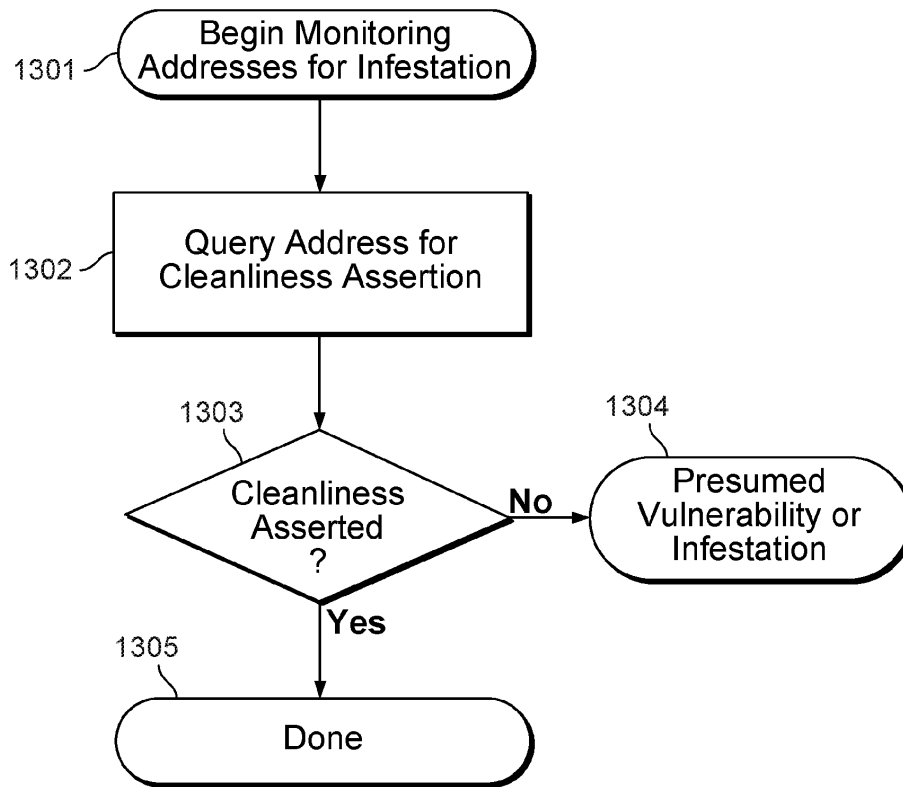


FIG. 13

Ex. 2, p. 19

K.Mizra LLC, Ex. 2019, p. 95

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 19 of 25

US 9,516,048 B1

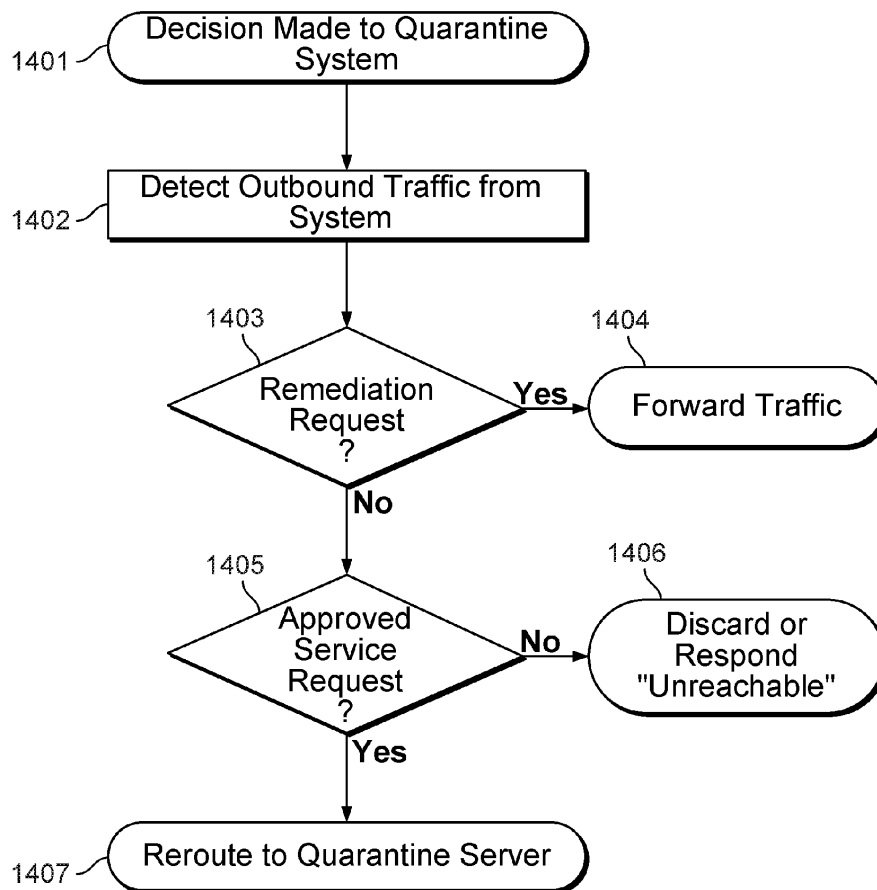


FIG. 14

Ex. 2, p. 20

K.Mizra LLC, Ex. 2019, p. 96

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 20 of 25

US 9,516,048 B1

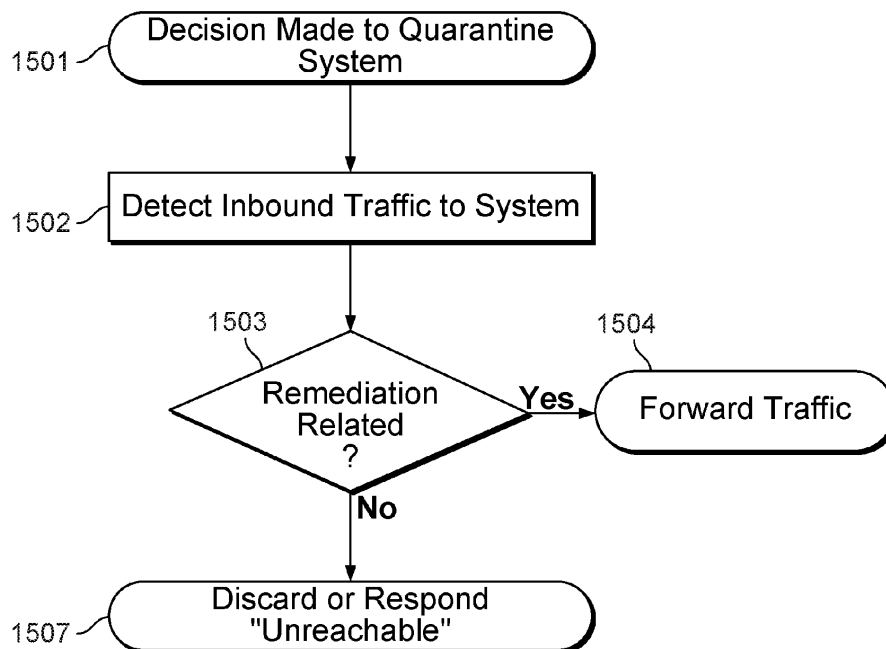


FIG. 15

U.S. Patent

Dec. 6, 2016

Sheet 21 of 25

US 9,516,048 B1

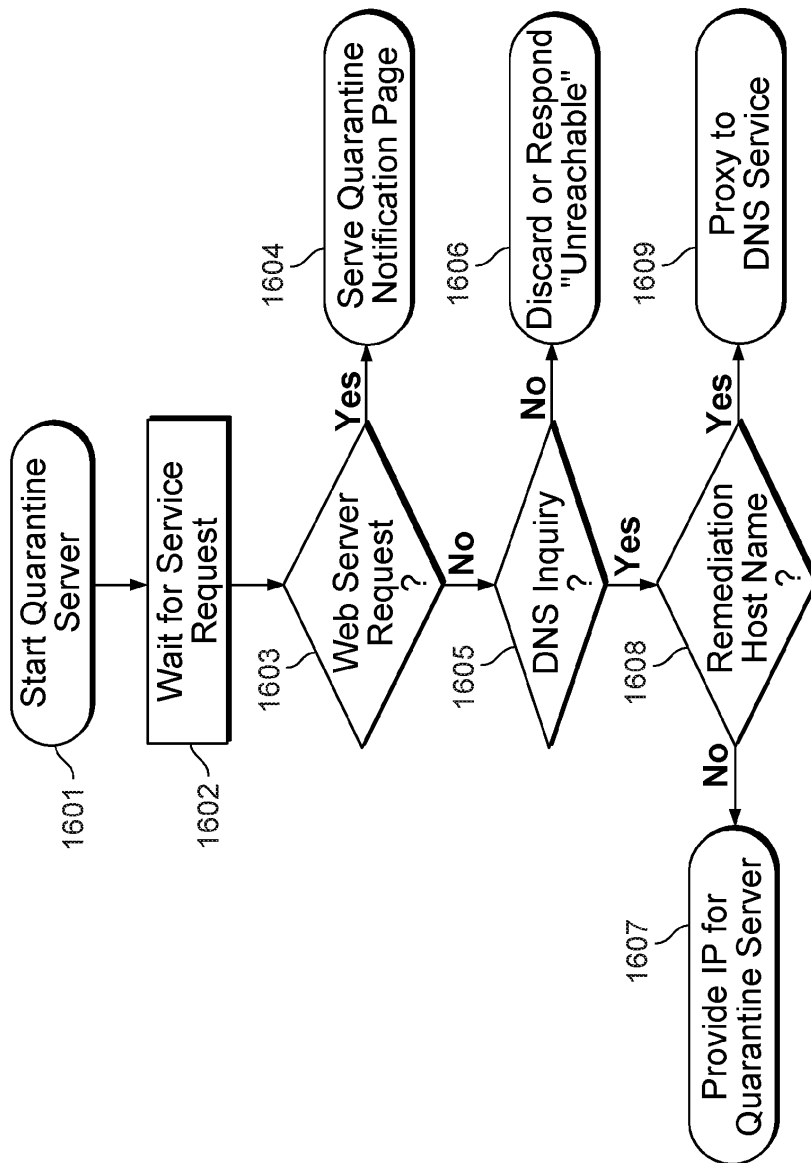


FIG. 16

Ex. 2, p. 22

K.Mizra LLC, Ex. 2019, p. 98

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 22 of 25

US 9,516,048 B1

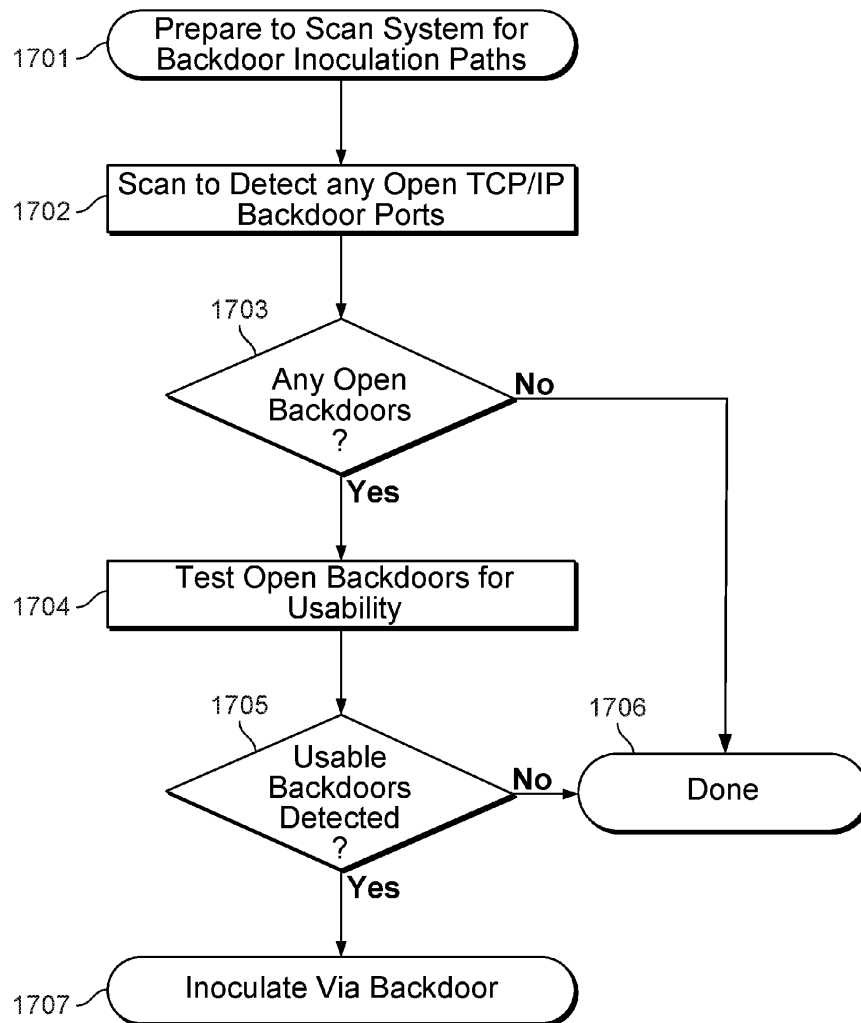


FIG. 17

Ex. 2, p. 23

K.Mizra LLC, Ex. 2019, p. 99

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 23 of 25

US 9,516,048 B1

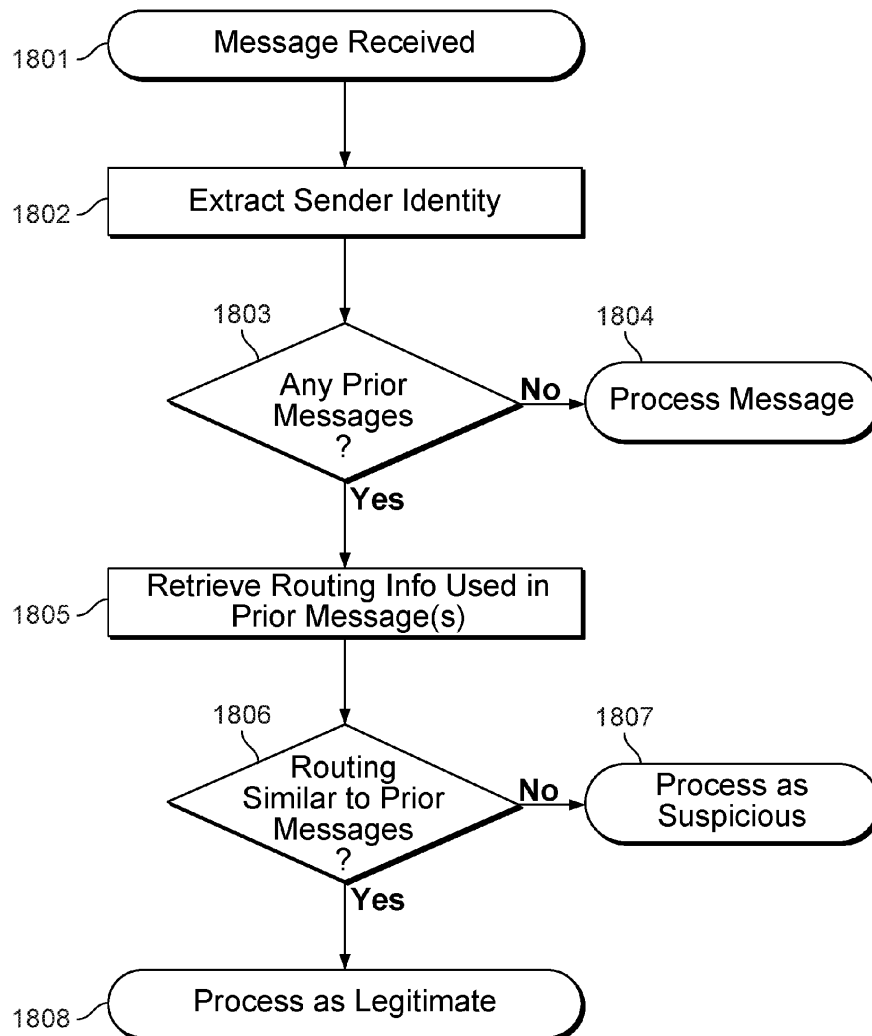


FIG. 18

Ex. 2, p. 24

K.Mizra LLC, Ex. 2019, p. 100

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 24 of 25

US 9,516,048 B1

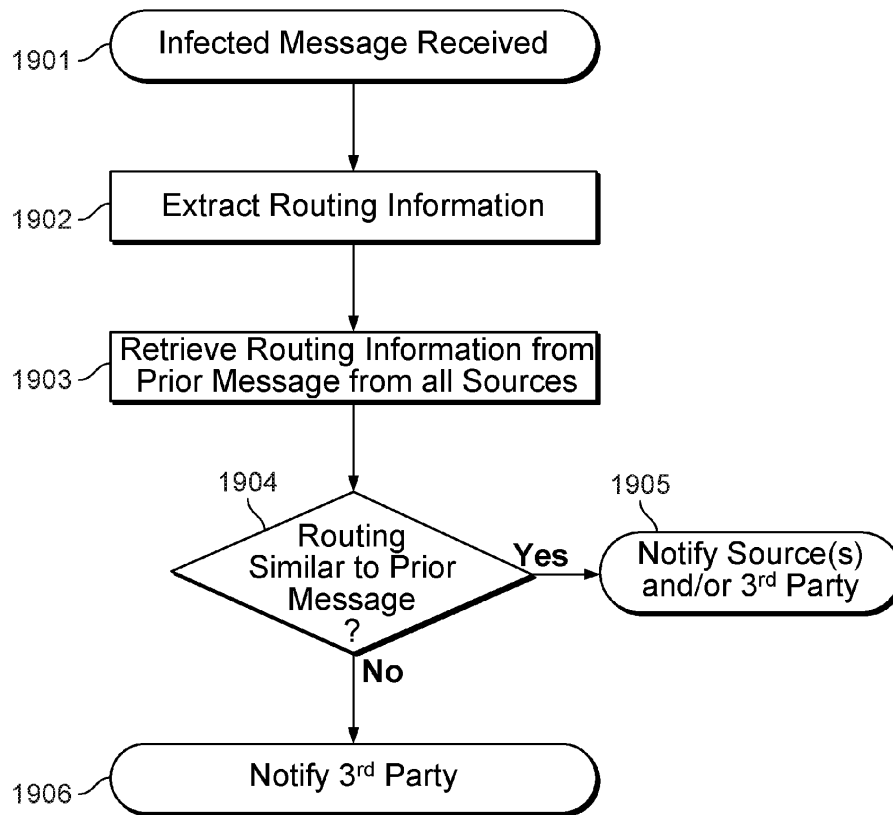


FIG. 19A

Ex. 2, p. 25

K.Mizra LLC, Ex. 2019, p. 101

Google LLC v. K.Mizra LLC

IPR2025-01436

U.S. Patent

Dec. 6, 2016

Sheet 25 of 25

US 9,516,048 B1

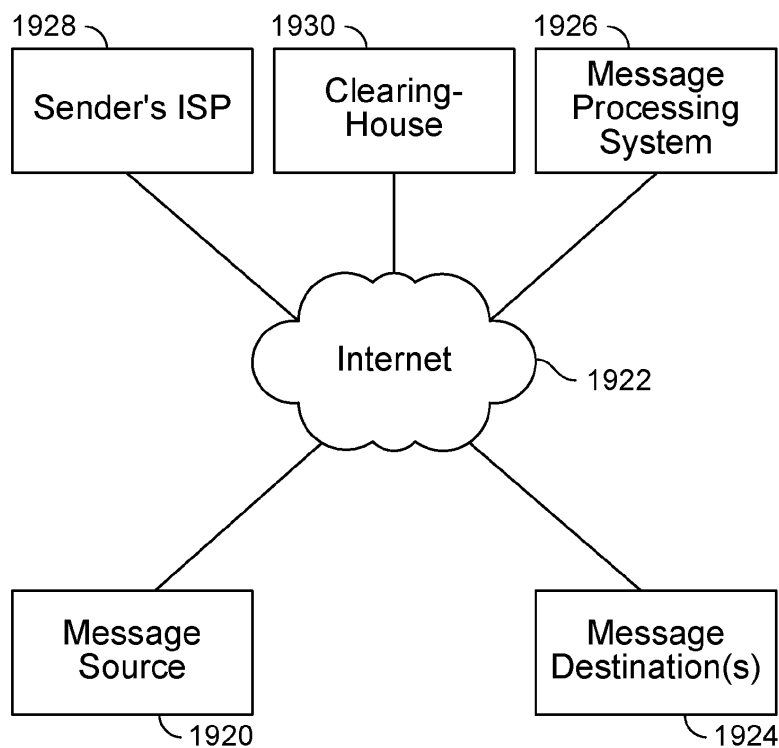


FIG. 19B

Ex. 2, p. 26

K.Mizra LLC, Ex. 2019, p. 102

Google LLC v. K.Mizra LLC

IPR2025-01436

US 9,516,048 B1

1

**CONTAGION ISOLATION AND
INOCULATION VIA QUARANTINE****CROSS REFERENCE TO OTHER
APPLICATIONS**

This application is a continuation of co-pending U.S. patent application Ser. No. 11/237,004, entitled Detecting and Processing Suspicious Network Communications, filed Sep. 27, 2005, which is incorporated herein by reference for all purposes, which claims priority to U.S. Provisional Patent Application No. 60/613,909 entitled CONTAGION ISOLATION AND INOCULATION filed Sep. 27, 2004, which is incorporated herein by reference for all purposes.

This application is related to U.S. patent application Ser. No. 11/237,003, entitled Contagion Isolation and Inoculation, filed Sep. 27, 2005, now U.S. Pat. No. 8,234,705.

BACKGROUND OF THE INVENTION

Unwanted and/or malicious network communications, such as spam, phishing, worm propagation, etc., hamper productivity and the use and enjoyment of computer and network resources by end users, burden affected networks with unauthorized and/or undesired traffic, and expose recipients to the risk of theft, fraud, etc. Current technologies for dealing with such unauthorized and/or unwanted communications focus on filtering messages at a point in the communication path close to the recipient system, such as anti-spam and/or other security software installed on the destination host and/or a local mail (or other messaging) server to which communications to the destination host are directed for delivery to the destination host, e.g., by detecting messages sent by blacklisted senders and/or messages containing header or other content that indicates the message is suspicious. While this approach can be an effective way to protect an individual host or a group of destination hosts served by a local server, it does not address the network congestion problem and leaves vulnerable hosts not protected by destination (or destination mail or messaging server) based filtering software. In addition, in many cases an attacker causes each of one or more compromised hosts to send multiple instances of a malicious communication, often simultaneously or nearly so, magnifying the effect on the network. Therefore, there is a need for an effective way to intercept and take corrective action with respect to unauthorized, unwanted, and/or otherwise malicious electronic mail and/or other network communications that better protects the network and provides protection to destination hosts that are not protected by destination or destination mail or messaging server based filtering software.

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

FIG. 2B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments.

2

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

FIGS. 3B-3D illustrate distributed update distribution as implemented according to some embodiments.

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

FIG. 4B is a block diagram of a client system configured to obtain an update according to some embodiments.

FIG. 4C is a block diagram of a system used to distribute 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. 10A is a flow diagram of a method for identifying and remedying problems according to some embodiments.

FIG. 10B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined 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 an infestation according to some embodiments.

FIG. 13 is a flow diagram of a method for monitoring one or more computers for infestation 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. 19A is a flow diagram of a method for notifying possible sources of infection according to some embodiments.

FIG. 19B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected.

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. A component such as a processor or a memory described as being configured to perform a task includes both a general component that is temporarily configured to perform the task at a given time and a specific component that is manufactured to perform

Ex. 2, p. 27

K.Mizra LLC, Ex. 2019, p. 103
Google LLC v. K.Mizra LLC
IPR2025-01436

US 9,516,048 B1

3

the task. 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 limited only by the claims 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 according to the claims 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.

Analyzing routing information to identify and intercept unauthorized, unwanted, and/or otherwise malicious communications is disclosed. In some embodiments, routing information associated with a message currently being processed is checked against corresponding information associated with a message from the same source that was processed previously. In some embodiments, a message processing system that handled and/or otherwise has access to routing information for both messages performs the check, such as a network messaging and/or access service provider with which the sender is associated, performs the check. If the routing information for the current message does not match the corresponding information observed in the previous message, responsive action is taken.

Contagion refers herein to software programs and/or code, data values, configurations, and settings, that expose a computer or other system to security threats, such as unauthorized access and/or use of the system and/or data stored thereon; corruption or loss of data; and/or damage, loss, and/or unauthorized reconfiguration of system software and/or hardware components. Examples of contagion include software programs and/or code designed to perform undesirable and/or unauthorized 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 refers herein to a 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

4

content such as extraordinarily long strings, or specific content detected to be malicious. Recent activity may be evaluated (102) to see if it is anomalous. Examples of anomalous activity include median email latency in mail queues beyond some threshold, for example above a 4 hour median queuing delay, and 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 anomalous (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. Another example of reducing the next update interval is to reduce the currently remaining time until an update, for example reducing by a proportion, such as 50%, the remaining time until an update is requested.

FIG. 2A 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 antivirus 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. 3A, which may include updating or concluding that an update is not needed. A time for next contacting an anti-contagion source is decided upon (204). In some embodiments, the scheduled time may be made sooner when the source of the update, such as a vendor of software to which the update pertains, 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. 2B is a block diagram of a network environment in which anti-contagion updates are distributed according to some embodiments. A plurality of clients 1 to "n", represented in FIG. 2B by clients 220, 222, and 224, are connected to the Internet 226. An update server 228 also is connected to the Internet 226. In various embodiments, one or more of the clients 220-224 is configured to request and obtain updates, if any, from update server 228 via the Internet, e.g., as described above in connection with FIG. 2A.

FIG. 3A 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 (301), for example as was discussed in 202 of FIG. 2A. Contact may be made with an

Ex. 2, p. 28

US 9,516,048 B1

5

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. 4A. After the update has been acquired (305), the contact may be complete (306).

Distributed update distribution is disclosed. In some embodiments, at least some users receive an update indirectly, e.g., from another user that received it previously, either directly from the original source or indirectly from a source that had itself earlier received it, again either directly or indirectly, and so on. FIGS. 3B-3D illustrate distributed update distribution as implemented according to some embodiments. In the example shown in FIGS. 3B-3D, it is assumed for expository purposes that each host that receives the update will serve as a redistribution node, each recipient has the ability to redistribute at the same rate as the original source, and there is continuous demand for the update from all sources. As shown in FIG. 3B, at a time t_1 a first direct recipient of the update D1 (332) downloads the update from the original source S1 (330). As shown in FIG. 3C, at a time t_2 a second direct recipient D2 (334) downloads the update from the original source and an indirect recipient R11 (336) downloads the update from the first direct recipient D1 (332), which in this case acts as a redistribution node. As shown in FIG. 3D, at a time t_3 a third direct recipient D3 (338) downloads the update from original source S (330), a second indirect recipient R12 (340) downloads the update from the first direct recipient D1 (332), a third indirect recipient R21 (342) downloads the update from the second direct recipient D2, and a fourth indirect recipient R111 (344) downloads the update from first indirect recipient R11 (336). In this example, in the three download cycles shown in FIGS. 3B-3D on its own the original source would only have been able to download the update to the three direct recipients, but by using previous recipients to redistribute, in the example shown in FIGS. 3B-3D a total of seven recipients have been able to obtain the update. In a generalized case, under the assumptions used in this example in "n" download cycles/periods a total of $2^n - 1$ recipients are able to obtain the update, whereas only "n" recipients, i.e., one

6

per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source. In an even more generalized case, under an assumption that each server may download roughly k copies in a time interval, for example where more than one download is provided by a server at a time, then in "n" download cycles (or periods) a total of roughly $(k+1)^n - 1$ recipients are able to obtain the update, where as only $k \cdot n$ recipients, i.e., k per cycle, would otherwise have been able to obtain the update if it were only available directly from the original source.

FIG. 4A is a flow diagram of a method for acquiring an update according to some embodiments. In this example, an update source is contacted (401), such as via an Internet connection, for example via TCP/IP. An offer to serve as a redistribution source is made (402). In some embodiments, 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. In various embodiments, 402 is optional, performed only if a criterion or test is satisfied, or omitted. An update or an update identifier is 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 confirms redistribution sourcing status (408) in this example. In some embodiments, confirmation includes 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 acquired (405) in this example. In some embodiments, the list is or may be explicit. In some embodiments, the list is or 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 are dynamic, and calculated or created in part on an as needed or as requested basis. In some embodiments, the list of other sources 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. In some embodiments, a requesting host that agrees to serve as a redistribution source is provided with a list of sources from which the update is available immediately, or at least sooner than it will be made available to hosts that will not redistribute. One or more 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 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 or within a recipient's intranet or home network. If a reservation time was acquired, then a wait for that reservation time (410) is performed in this example. At the reservation time, a further attempt to acquire

Ex. 2, p. 29

US 9,516,048 B1

7

an update or update identifier from the source (403) is made, and the flow proceeds as described earlier. This flow may continue until an acquisition is complete (408), as also described earlier. In various embodiments, one or more of confirming redistribution sourcing status (408), attempting to obtain the update from a redistribution source (405-407), and/or acquiring and/or waiting for a reservation time (409-410) is/are optional, conditional, or omitted.

FIG. 4B is a block diagram of a client system configured to obtain an update according to some embodiments. In some embodiments, the client system shown in FIG. 4B is used to implement one or more of clients 220-224 of FIG. 2B. The client system 420 includes a communication interface 422, such as a network interface card, configured to send/receive communications via a network, such as the Internet. The client system 420 further includes an update agent and/or browser 424. In some embodiments, the update agent/browser 424 comprises browser software used to navigate the World Wide Web, e.g., to send/receive data to/from a website from which an update is available for download. In some embodiments, update agent/browser 424 includes an agent associated with anti-contagion software installed on the client system 420. In various embodiments, the update agent/browser 424 comprises one or more components and is implemented as one or more software programs running on a processor and/or one or more hardware components. A settings data store 426 stores one or more settings and/or other data associated with the software with which an update and/or software to which the update does or may pertain, such as periodicity or next scheduled (e.g., reservation) time to check for updates, a list of update identifiers, locators, and/or redistribution sources from which an update is or may be available, policies for whether and/or under what terms, conditions, circumstances, etc. the client system 420 may be used redistribute an update, a last update time, an old version identifier (see, e.g., 202 of FIG. 2A), etc.

FIG. 4C is a block diagram of a system used to distribute an update according to some embodiments. In some embodiments, the system of FIG. 4C is used to provide an update server, such as update server 228 of FIG. 2B. In this example, the distribution system 440 includes a communication interface 442 configured to send/receive data via a network, such as the Internet. An update distribution engine 444 is configured to receive and service requests for updates. For updates provided directly to a request source, the update distribution engine 444 obtains the update from an update storage location 446 and provides the update to the request source via the communication interface 442. In some embodiments, if data included in the request indicates the source of the request is or may be willing to serve as a redistribution source, the update distribution engine 444 hands the request off to a redistribution coordinator 448, which negotiates with the request source, e.g., based on policies, settings, rules, etc. stored in a policies and lists data store 450, through communications sent/received via communication interface 442, to determine if the requesting source will serve as a redistribution source and, if so, to determine the terms and conditions of such service and data, such as an address, locator, or identifier, that can be provided to other requesting systems and/or users to enable them to attempt to obtain the update from the redistribution source. In some embodiments, if update distribution engine 444 determines that it is not able to provide the update to the request source directly (e.g., due to capacity limits or because a policy applicable to the request and/or the request source indicates the request source should be directed to a

8

redistribution source, e.g., because the request source has a less expensive version of the software or is not willing to serve as a redistribution source), the update distribution engine 444 hands the request off to redistribution coordinator 448, which provides to the request source an update identifier and a list of one or more redistribution sources from which the update is or may be available to the request source.

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 some embodiments, a multicast 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 a provided 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 a 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/or 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

Ex. 2, p. 30

US 9,516,048 B1

9

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 a user cannot modify.

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 a request for content. The redistribution computer may optionally contact another computer, for example a more original source, and affirm its redistribution sourcing status (702). A

10

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 demonstrates that the redistribution computer is performing redistribution. In some embodiments, contacting a more original source allows 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. An authentication to be used to authenticate requests is negotiated (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. 4A. In one example, an indirect source may be a peer, such as another computer that has downloaded the content. In another example, an indirect 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 or authorized 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. A next address and (optional) authenticator in the list of sources is retrieved (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/or higher bandwidth, for example by first processing a local address on a LAN. In some embodiments, elements of the list are or 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), which may in some embodiments be performed as described in conjunction with 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 a connection such as a TCP/IP connection with the source. If the acquisition is complete (804), then the processing is complete (805) in this example. If the acquisition 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 process-

Ex. 2, p. 31

US 9,516,048 B1

11

ing 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. Examples of a server include an HTTP server, an FTP server, and a multicast server. A server waits 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 receiving and validating, or performing a mutual proof of possession of, 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 considered 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/or an appropriate coding, such as uuencode.

FIG. 10A 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 presence and/or 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 FIGS. 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 the problem is quarantined (1002). In some embodiments, quarantining includes preventing 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 are authorized. In some embodiments, quarantining includes preventing undesired communications from reaching the quarantined computer, such as control messages or infectious messages. Examples of quarantining activities are illustrated in FIGS. 14 and 15. In this example, the quarantined computer is provided access to remediation services (1003). In some embodiments, access to remediation includes or 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 includes allowing

12

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 is restricted by the quarantine to 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. 10B is a block diagram illustrating a network environment in which infected hosts and/or networks are quarantined according to some embodiments. A plurality of hosts 1 to "m", represented in FIG. 10B by hosts 1020, 1022, and 1024, connect via a router 1026 and a gateway 1028 to the Internet 1030. In various embodiments the router 1026 is omitted or included in the gateway 1028. In various embodiments, gateway 1028 comprises a gateway, router, firewall, or other device configured to provide and control access between a protected network and the Internet and/or another public or private network. In various embodiments, the process of FIG. 10A is implemented, in the event of quarantine of one or more of hosts 1020-1024, for example, on the affected (quarantined) host, router 1026, and/or gateway 1028. In some embodiments, a quarantined host (or a host associated with a quarantined network or sub-network) is permitted to access a remediation server 1032, e.g., to download a patch, more current threat definition, etc. In some embodiments, requests to connection to a host other than the remediation server 1032 are redirected to a quarantine server 1034 configured to provide a notice and/or other information and/or instructions to a user of the quarantined host. In various alternative embodiments the quarantine server 1034 is omitted or resides on the router 1026 and/or gateway 1028.

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 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 scanning 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 no open ports are detected (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 is performed or attempted (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

Ex. 2, p. 32

US 9,516,048 B1

13

response when a vulnerability is present, such as sending a long string known to induce a buffer overflow with a consequent errant response, where the errant response may for example include a characteristic communication, 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 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, or for example by extracting a MAC address from traffic associated with the IP address, 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 an infestation according to some embodiments. In this example, a device near a computer, such as a firewall on a LAN or an ISP's router, is used in some embodiments to monitor a computer for infestation (1201). Network traffic is monitored for anomalous traffic volume, port, or IP destination (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 a 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 inoculation, 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 num-

14

ber 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 one or more computers for infestation according to some embodiments. In this example monitoring of a 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. A computer associated with an address identified in a list is 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 a computer is the Paladium security initiative under development by Microsoft and supported by Intel and American Micro Devices. Another example of a trusted computing base is described in various TCG specifications, such as the TCG Architecture Overview, published by the Trusted Computing Group. Trusted code bases may for example execute antivirus scans of the remainder of the computer, including untrusted portions of the disk and/or operating system. In some embodiments, trusted code bases may digitally sign assertions about the cleanliness (e.g. infestation status) and/or 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 a version associated with a current anti-contagion software or definition file in use, wherein a sufficiently updated software and/or scan may act as a cleanliness assertion. In some embodiments, an operating system may respond with information associated with its patch level, wherein a sufficiently recent patch level may be interpreted as an assertion of cleanliness. If a computer asserts it is clean (1303) then monitoring is 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

Ex. 2, p. 33

US 9,516,048 B1

15

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 is 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 is tested in this example 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, such as an internal ISP site. If outbound traffic is deemed to be associated with a remediation request (1403), then the traffic is 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 request is to respond with a redirect message, which would direct a browser on a quarantined computer to contact a quarantine server that provides remediation information. If the request was not for an approved service (1406), then the outbound traffic is discarded or responded to with an 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 of FIG. 14. Inbound traffic originally destined for a quarantined computer is 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 is tested in this example 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. In various embodiments, at least certain types of traffic from approved sites such as anti-contagion vendors, security update providers, and/or internal ISP sites are 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 an installation setting

16

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, a device such as a router may forward outbound traffic from a quarantined computer to a quarantine server, for example as illustrated in conjunction with 1407 of 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, and/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 is 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 partnered 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. A short time to live ensures that a further DNS query will be sent prior to subsequent communications being sent from the quarantined computer to the host that was the subject of the DNS query, aiding a return to normal routing of traffic once the computer is no longer under quarantine. In some embodiments, the server(s) in this example may con-

Ex. 2, p. 34

US 9,516,048 B1

17

tinue 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 information associated with pre-loaded 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. In some embodiments, a user is requested and/or required to authorize in advance use of an authorized and/or any unauthorized backdoor for purposes of inoculation. 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 one or more backdoor ports are open (1703) then they are tested for usability (1704) in this example. In some embodiments, a backdoor port may be assumed open (1703) and 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 as a buffer overrun exploit that does not have a permanent impact but provides observable validation (such as a dropped connection). If a backdoor 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

18

operations. Another example of inoculation includes a combination of one or more of the above techniques, along with an optional 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 is extracted (1802), for example by examining the "From:" header line in an email message, or by examining the envelope information conveying an email message, or by examining the sender's ID in an instant message. A check is 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 is 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 information for a message purporting to be from a sender is not similar to previous messages from the same sender (1806) then the message is 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

Ex. 2, p. 35

US 9,516,048 B1

19

a user agree to ignore warnings which describe the suspicious circumstances. In various embodiments, the process of FIG. 18 is implemented on a mail, messaging, or other server associated with the sender and/or the sender's ISP; a mail, messaging, or other server associated with the recipient and/or the recipient's ISP; and/or one or more intermediate nodes.

FIG. 19A 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 as such by anti-contagion software. Routing information is 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 is 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 is compared with the retrieved routing information (1904). If no similarity with previous messages is found (1904) then in this example a third party notification is made (1906). 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 service that determines the credibility of a report, such as a collaborative filter service, that accumulates accusations of viral disseminations. 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 infected message source. If a similarity is found with previous messages (1904) then in this example a notification is sent (1905). 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.

FIG. 19B is a block diagram illustrating a network environment in which suspicious messages are identified and/or one or more notifications are sent when a malicious message is detected. A message source 1920 sends a message via the Internet 1922 to a message processing system 1926 for delivery via the Internet to one or more message destinations 1924. In various embodiments, the message processing system 1926 comprises a mail or messaging server associated with the message source 1920, a mail or messaging server associated with the destination(s) 1924, and/or an intermediate processing node. In some embodiments, the message source 1920 is connected directly and/or via a local network, and not via the Internet 1922 as shown in FIG. 19B, to message processing system 1926. In some embodiments, message processing system 1926 is configured to identify

20

suspicious messages using current and historic routing information, e.g., as described above in connection with FIG. 18. In some embodiments, the process of FIG. 19A is implemented on message processing system 1926. For example, in some embodiments if message processing system 1926 determines that a message sent by message source 1920 is infected, message processing system 1926 compares routing information associated with the malicious message with corresponding information for one or more messages received previously from the message source 1920. If the routing information is similar, in some embodiments the message processing system 1926 notifies the message source 1920. If the routing information is not similar, in various embodiments the message processing system 1926 notifies the message source 1920, the sender's ISP 1928, and/or a clearinghouse or similar applicable service 1930, as described above.

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.

What is claimed is:

1. A method, comprising:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

2. A method as recited in claim 1, wherein detecting the insecure condition further includes at least one of the group consisting of scanning for a vulnerability, scanning for

Ex. 2, p. 36

US 9,516,048 B1

21

malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

3. A method as recited in claim 1, wherein detecting the insecure condition includes determining that the first host should be quarantined until an update to an operating system has been installed.

4. A method as recited in claim 1, wherein permitting the first host to communicate with the remediation host includes:

detecting an outbound communication from the first host; and
forwarding the outbound communication if it is addressed to the remediation host.

5. A method as recited in claim 1, wherein quarantining the first host further includes preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

6. A method as recited in claim 1, performed at an Internet service provider.

7. A method as recited in claim 1, wherein the software component on the first host is an operating system.

8. A method as recited in claim 1, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

9. A method as recited in claim 8, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

10. A system, comprising:

a processor configured to:

detect an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantine the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that

22

causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and

permit the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition; and

a memory coupled to the processor and configured to provide instructions to the processor.

11. A system as recited in claim 10, wherein the processor is configured to detect an insecure condition at least in part by performing one or more of the following: scanning for a vulnerability, scanning for malicious data, determining whether a security software is installed, and detecting anomalous network traffic.

12. A system as recited in claim 10, wherein the processor is configured to detect an insecure condition at least in part by determining that an initial startup after installation of an operating system is being performed.

13. A system as recited in claim 10, wherein the processor is configured to quarantine the first host at least in part by preventing the first host from receiving via the protected network data not related to remediation of the insecure condition.

14. A system as recited in claim 10, wherein the software component on the first host is an operating system.

15. A system as recited in claim 10, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated.

16. A system as recited in claim 15, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent.

17. A computer program product, the computer program product being embodied in a non-transitory computer readable medium and comprising computer instructions for:

detecting an insecure condition on a first host that has connected or is attempting to connect to a protected network, wherein detecting the insecure condition includes contacting a trusted computing base associated with a trusted platform module within the first host, receiving a response, and determining whether the response includes a valid digitally signed attestation of cleanliness, wherein the valid digitally signed attestation of cleanliness includes at least one attestation selected from the group consisting of an attestation that the trusted computing base has ascertained that the first host is not infested, and an attestation that the trusted computing base has ascertained the presence of a patch or a patch level associated with a software component on the first host;

when it is determined that the response does not include a valid digitally signed attestation of cleanliness, quarantining the first host, including by preventing the first host from sending data to one or more other hosts associated with the protected network, wherein preventing the first host from sending data to one or more other hosts associated with the protected network includes receiving a service request sent by the first host, determining whether the service request sent by the first host is associated with a remediation request, and when it is determined that the service request sent by the first host is not associated with a remediation request, serving a quarantine notification page that provides remediation information to the first host if the service request sent by the first host comprises a web

Ex. 2, p. 37

K.Mizra LLC, Ex. 2019, p. 113

Google LLC v. K.Mizra LLC

IPR2025-01436

US 9,516,048 B1

23

server request, wherein serving the quarantine notification page to the first host includes re-routing by responding to the service request sent by the first host with a redirect that causes a browser on the first host to be directed to a quarantine server configured to serve the quarantine notification page; and
5 permitting the first host to communicate with a remediation host configured to provide data usable to remedy the insecure condition.

18. A computer program product as recited in claim 17, wherein the software component on the first host is an operating system. 10

19. A computer program product as recited in claim 17, wherein determining that the response does not include a valid digitally signed attestation of cleanliness includes determining that the software component on the first host is not sufficiently updated. 15

20. A computer program product as recited in claim 19, wherein determining that the software component on the first host is not sufficiently updated includes determining that a patch level associated with the software component on the first host is not sufficiently recent. 20

* * * * *

24

EXHIBIT 3

Chrome Enterprise Premium documentation

View all product documentation (<https://cloud.google.com/beyondcorp-enterprise/docs/overview>)

Chrome Enterprise Premium is Google Cloud's zero-trust solution that enables an organization's workforce to access web applications securely from anywhere, without the need for VPN and without fear of malware, phishing, and data loss.

By using Chrome Enterprise Premium, you can manage access for apps on Google Cloud, other clouds, and on-premises, define and enforce access policies based on user, device, and other contextual factors, and make apps more accessible and responsive through Google's global network.

Keep exploring with 20+ always-free products

Access 20+ free products for common use cases, including AI APIs, VMs, data warehouses, and more.

View free product offers (</free/docs/free-cloud-features#free-tier-usage-limits>)

Documentation resources

Find quickstarts and guides, review key references, and get help with common issues.



Guides

[Quickstart: Apply Chrome Enterprise Premium to cloud resources](#)

(/beyondcorp-enterprise/docs/apply-resources)

[Overview](#)

(/beyondcorp-enterprise/docs/overview)

[Secure the Google Cloud console and APIs](#)

(/beyondcorp-enterprise/docs/securing-console-and-apis)



Reference

[Audit logging](#)

(/beyondcorp-enterprise/docs/audit-logs)

[Using hostname and path conditions](#)

(/beyondcorp-enterprise/docs/hostname-and-path-conditions)



Resources

[Pricing](#)

(/beyondcorp-enterprise/pricing)

[Release notes](#)

(/beyondcorp-enterprise/docs/release-notes)

[Support](#)

(/beyondcorp-enterprise/docs/support)

[Billing questions](#)

(/beyondcorp-enterprise/docs/billing-questions)

[Securing on-premises apps and resources with IAP](#)

(/beyondcorp-enterprise/docs/securing-on-premises)

[Securing App Engine apps with IAP](#)

(/beyondcorp-enterprise/docs/securing-app-engine)

[Securing non-Google Cloud applications using the app connector](#)

(/beyondcorp-enterprise/docs/enable-app-connector)

Related resources

Explore self-paced training from Google Cloud Skills Boost, use cases, reference architectures, and code samples with examples of how to use and connect Google Cloud services.

TRAINING

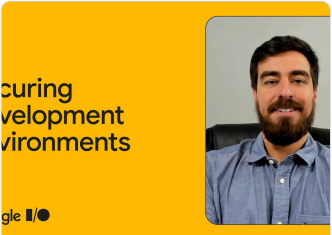
Security in Google Cloud

In this training course, you will learn about a variety of Google Cloud security controls and techniques. You'll explore the components of Google Cloud and deploy a secure solution on the platform. You'll also learn how to mitigate attacks at several points in a Google Cloud-based infrastructure, including distributed denial-of-service attacks, phishing attacks, and threats involving content classification and use.

Lear
n
mor
e

[\(/training/course/security-in-google-cloud-platform\)](/training/course/security-in-google-cloud-platform)

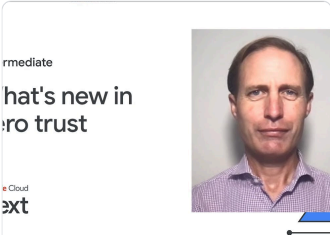
Related videos



YOUTUBE

[Securing development environments with Google Cloud](https://www.youtube.com/watch?v=hVTnW-rH5Vc)

(<https://www.youtube.com/watch?v=hVTnW-rH5Vc>)



YOUTUBE

[What's new in zero trust](https://www.youtube.com/watch?v=KaeL0C39jDQ)

(<https://www.youtube.com/watch?v=KaeL0C39jDQ>)



YOUTUBE

[Google Cloud Security Summit](https://www.youtube.com/watch?v=3kMKKaVGQZk)

(<https://www.youtube.com/watch?v=3kMKKaVGQZk>)

Software supply chain security starts where code is created, in the development environments/IDEs. Learn ho...

Google pioneered zero trust for corporate applications with its BeyondCorp approach, and then combined it with in-...

The Google Cloud Security Summit brought together leading Google and partner professionals to present the...

[More](#) ▾

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-01-09 UTC.

EXHIBIT 4

Endpoint Verification overview

This document describes the basic concepts of Endpoint Verification.

Endpoint Verification lets security administrators or security operations professionals secure Google Cloud, on-premises apps and resources, and Google Workspace apps.

Endpoint Verification is part of Google Cloud [Chrome Enterprise Premium](#) (/beyondcorp-enterprise/docs/overview) and is available to all Google Cloud, Cloud Identity, Google Workspace for Business, and Google Workspace for Enterprise customers.

When to use Endpoint Verification

Use Endpoint Verification when you want an overview of the security posture of the devices that are used to access your organization's resources, such as laptops and desktops.

As a security administrator or security operations professional, your goal is to manage secure access to your organization's resources. The employees of your organization can use either the company-owned devices or their unmanaged personal devices to access the organization's resources. When Endpoint Verification is installed on the devices that access your organization's resources, it collects and reports device inventory information. You can use this device inventory information to manage secure access to your organization's resources.

When paired with the other offerings of Chrome Enterprise Premium, Endpoint Verification helps enforce fine-grained access control on your Google Cloud resources.

How Endpoint Verification works

Endpoint Verification consists of a Chrome extension that collects and reports device inventory information by constantly syncing with Google Cloud. Endpoint Verification creates an inventory of devices with Chrome browser that access your organization's data.

For example, after Endpoint Verification is deployed on devices that are used to access Google Cloud resources, Endpoint Verification populates information about those devices. As an administrator, you can review the device information including encryption status, OS, and other details, and use this information to manage access control.

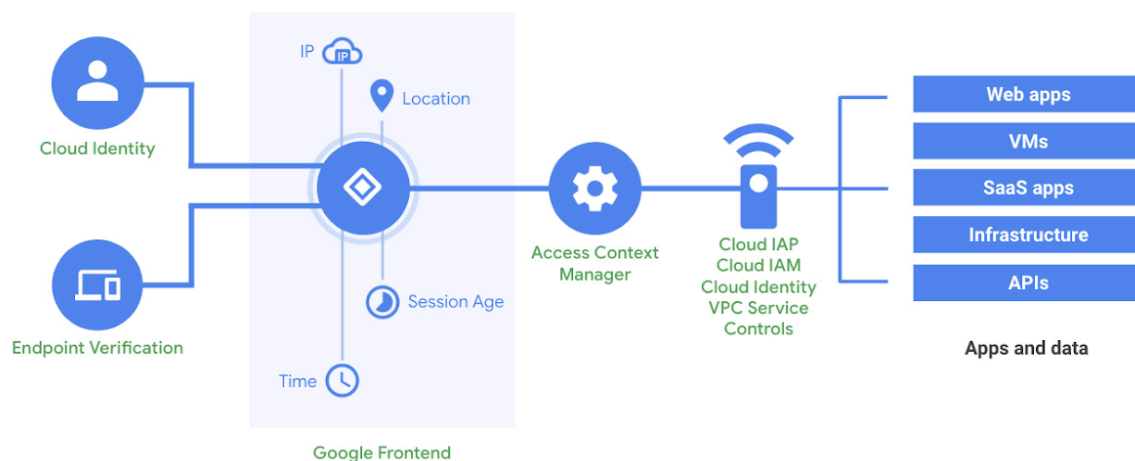
For more information, see [Device attributes collected by Endpoint Verification](#) (/endpoint-verification/docs/device-information).

How Endpoint Verification works with Access Context Manager

[Access Context Manager](#) (/access-context-manager/docs), which is part of Google Cloud [Chrome Enterprise Premium](#) (/beyondcorp-enterprise/docs/overview), lets security administrators or security operations professionals define fine-grained and attribute-based access control for projects and resources in Google Cloud and resources in Google Workspace.

Access Context Manager references the device attributes collected by Endpoint Verification to enforce fine-grained access control with [access levels](#) (/access-context-manager/docs/overview#access-levels).

The following diagram shows how Endpoint Verification works with Access Context Manager:

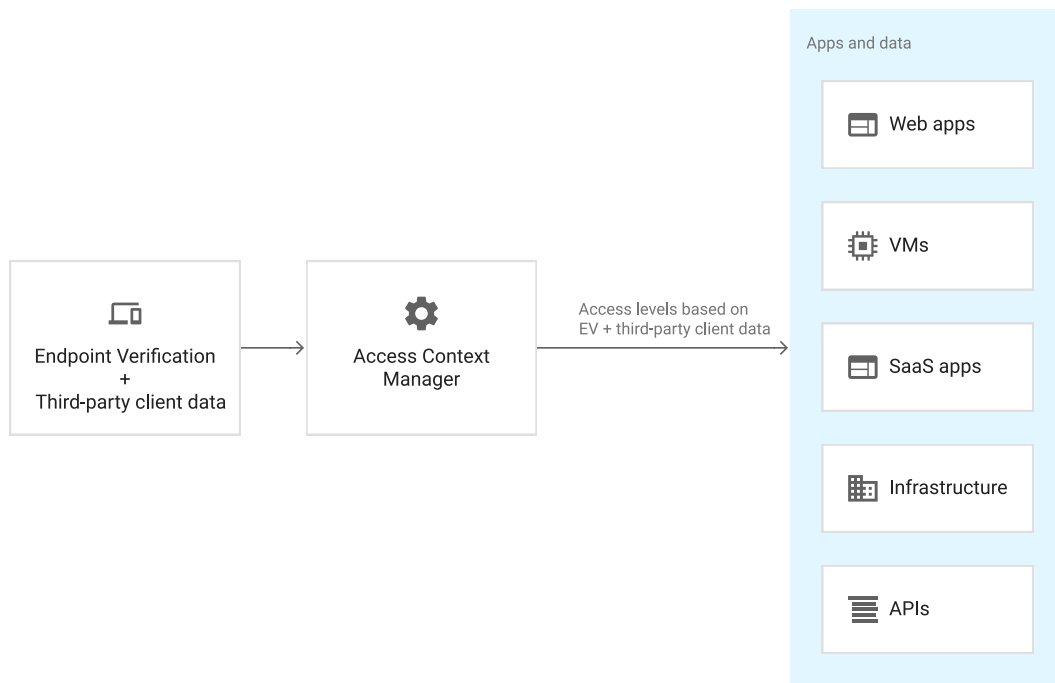


How Endpoint Verification works with third-party clients

In [Chrome Enterprise Premium and third-party client integrations](#) (/beyondcorp-enterprise/docs/integration-docs), third-party clients such as CrowdStrike and Microsoft Intune collect real-time device information. Endpoint Verification communicates with these third-party clients to collect their device information and makes them available for [Access Context Manager](#) (/access-context-manager/docs).

Access Context Manager references the device attributes collected by Endpoint Verification and third-party clients to enforce fine-grained access control with access levels (</access-context-manager/docs/overview#access-levels>).

The following diagram shows how Endpoint Verification and third-party clients work with Access Context Manager:



What's next

- [Quickstart: Set up Endpoint Verification on your devices](/endpoint-verification/docs/quickstart) (/endpoint-verification/docs/quickstart)
- [Device attributes collected by Endpoint Verification](/endpoint-verification/docs/device-information) (/endpoint-verification/docs/device-information)
- [Deploy Endpoint Verification](/endpoint-verification/docs/deploying-with-admin-console) (/endpoint-verification/docs/deploying-with-admin-console)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

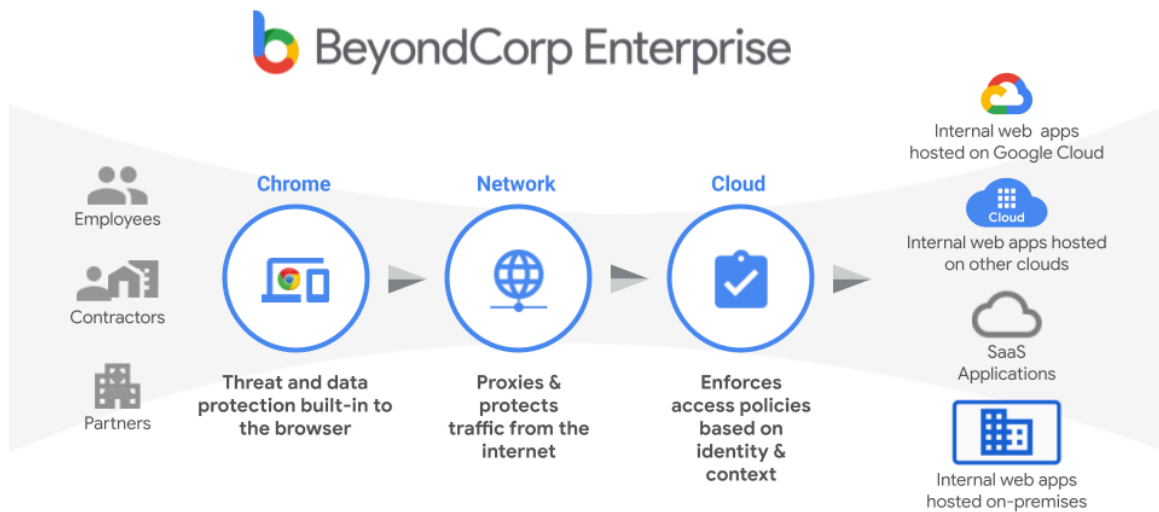
Last updated 2025-01-30 UTC.

EXHIBIT 5

Chrome Enterprise Premium overview

Today's enterprises are moving to a model of security where secured networks aren't enough. A modern approach is required to truly protect a company's most secure assets and allow for your employees to be productive under the right circumstances.

Chrome Enterprise Premium is Google's tooling meant to empower organizations to enable this new approach. By tying together a user's information with device and location context, an enterprise can make rich access decisions and enforce security policy.



Chrome Enterprise Premium has two key goals:

- Threat and data protection brings security to your enterprise devices by working to protect users from exfiltration risks such as copy and paste, extending DLP protections into the browser, and helping to prevent malware from getting onto enterprise-managed devices.
- Richer access controls protect access to secure systems (applications, virtual machines, APIs, and so on) by using the context of an end-user's request to ensure each request is authenticated, authorized, and as safe as possible.

Benefits to users

Chrome Enterprise Premium presents a security model that allows for greater security posturing and policy for both applications and devices, while providing end users a better user experience no matter where they access from or what type of device they use to do so:

- For administrators:
 - Strengthen security posture to account for dynamic changes in a user's context.
 - Shrink the access perimeter to only those resources that an end user should be accessing.

- Enforce device security postures for employees, contractors, partners, and customers for access, no matter who manages the devices.
- Extend security standards with per-user session management and multifactor authentication.
- For end users:
 - Allow all end users to be productive everywhere without compromising security.
 - Allow the right level of access to work applications based on their context.
 - Unlock access to personally-owned devices based on granular access policies.
 - Access internal applications without being throttled by segmented networks.

Common use cases

As end users work outside of the office more often and from many different types of devices, enterprises have common security models they are looking to extend to all users, devices, and applications:

- Allow non-employees to access a single web application deployed on Google Cloud or other cloud services platforms without requiring the use of a VPN.
- Allow non-employees to access data from their personal or mobile devices as long as they meet a minimum security posture.
- Ensure employees are prevented from copying and pasting sensitive data into email or saving data into personal storage such as Google Drive.
- Only allow enterprise-managed devices to access certain key systems.
- Provide DLP protections for corporate data.
- Gate access based on a user's location.
- Protect applications in hybrid deployments that use a mix of Google Cloud, other cloud services platforms, or on-premises resources.

Common signals

Chrome Enterprise Premium offers common signals enterprises can take into account when making a policy decision, including:

- User or group information
- Location (IP or geographic region)
- Device
 - Enterprise-managed devices
 - Personally-owned devices
 - Mobile devices
- Third-party device signals from partners in the BeyondCorp Alliance.
 - Check Point

- CrowdStrike
- Lookout
- Tanium
- VMware

Note: Google is not responsible for the accuracy of device data generated by third-party partners. Data provided to Google by the third-party partner is stored as-is. Any inaccuracies or personally identifiable information (PII) reported by the third party partner are the sole responsibility of the partner.

- Risk scores

How to get Chrome Enterprise Premium

[Complete this form](https://inthecloud.withgoogle.com/beyondcorp/contact.html) (<https://inthecloud.withgoogle.com/beyondcorp/contact.html>) to get more information about upgrading to Chrome Enterprise Premium.

Chrome Enterprise Premium compared with Google Cloud

Chrome Enterprise Premium provides enterprise security features in addition to the basic protections, focused on protecting applications with authentication and authorization, that are baseline features of Google Cloud. Chrome Enterprise Premium extends those protections to applications and data running everywhere, with end-user protections and rich access policy protections.

The following table shows the differences between the baseline features available to Google Cloud customers and what is available with Chrome Enterprise Premium:

Applications and Resources Access	GCP Baseline	BeyondCorp Enterprise Essentials	BeyondCorp Enterprise
Access control to web applications on Google Cloud Platform	✓		✓
Access control to SSH, RDP and TCP ports for VMs on GCP	✓		✓
Access control to Google Cloud Platform APIs	✓		✓
Access control to Google Cloud console	✓		✓
Access control to web applications on GCP internal load balancing	✓		✓
Access control to web applications on customer premises			✓
Access control to thick client / client-server applications			✓
Access control to web applications on AWS and Azure			✓
Access control to SAML-based applications (login time)		✓	✓
Access control to Google Workspace Admin Console		✓	✓
Access Policies and Advanced Settings	GCP Baseline	BeyondCorp Enterprise Essentials	BeyondCorp Enterprise

Access levels using users	✓	✓	✓
Access levels using IP addresses and geolocations	✓	✓	✓
Access levels using time and date restrictions		✓	✓
Access levels using login credential strength		✓	✓
Access levels using enterprise certificates		✓	✓
Access levels using device security postures		✓	✓
Access levels using Chrome security postures		✓	✓
Access levels using third party partner signals		✓	✓
Access levels using advanced expression language		✓	✓
Same-origin policy configuration in HTTP OPTIONS	✓	N/A	✓
Custom authentication domain and 403 pages		N/A	✓
User, Threat, and Data Protection	GCP Baseline BeyondCorp Enterprise Essentials BeyondCorp Enterprise		
Data loss prevention w/ predefined or custom detectors (Chrome)		✓	✓
Malware protection w/ advanced sandboxing (Chrome)		✓	✓
Phishing and malicious URL protection (Chrome)		✓	✓
Threat and data protection alerting and reporting (Chrome)		✓	✓

What's next

- Learn more about [access protection controls](#) (/beyondcorp-enterprise/docs/access-protection)
- Learn more about [threat and data protections](#) (https://support.google.com/a/answer/10104463)

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](#) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](#) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](#) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-01-08 UTC.

EXHIBIT 6

Security & Identity

How to prevent account takeovers with new certificate-based access

September 17, 2024

Ratan Nalumasu

Director, Engineering, Google Cloud

Christopher Altman

Sr. Product Manager, Google Cloud

Stolen credentials are one of the [top attack vectors](#) used by attackers to gain unauthorized access to user accounts and steal information. At Google, we're continually evolving security capabilities and practices to make our cloud the most trusted cloud. To help protect your organization from stolen credentials, cookie theft, and accidental credential loss, we're excited to announce the general availability of [certificate-based access](#) in our [Identity and Access Management](#) portfolio.

Certificate-based access (CBA) uses [mutual TLS](#) (mTLS) to ensure that user credentials are bound to a device certificate before authorizing access to cloud resources. CBA provides strong protection requiring [X.509 certificates](#) as device identifiers, and verifies devices with user context for every access request to cloud resources. Even if an attacker compromises a user's credentials, account access will remain blocked as they do not have the corresponding certificate. This renders the stolen credentials useless.

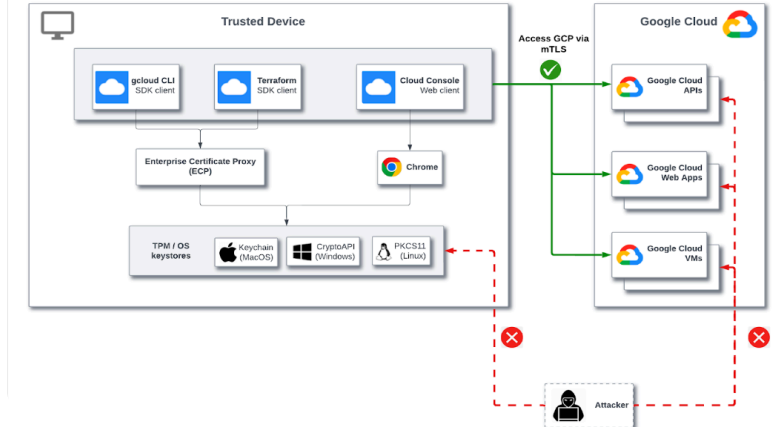
Mitigating account takeovers with certificate-based access

Certificate-based access can help you build a multi-layered defense against account takeovers, bolster data security, and maintain user trust. At Google, we have used this [unique Zero Trust approach](#) for many years as a strong defense to protect our technical infrastructure and employees. Now with CBA, we are extending this same level of security to our Google Cloud customers. Here are the key attributes of this approach:

- **Certificate-based access control:** Granular access policies with X.509 device certificates ensures only legitimate users with the correct certificate can access cloud resources.
- **Protections beyond initial login:** In contrast to using mTLS only for authentication, CBA also evaluates every authorization request to help safeguard resource access.
- **Strong key protection:** CBA's use of mTLS leverages secure cryptographic storage such as [TPMs and OS keystores](#). Tooling such as [Enterprise Certificate Proxy](#) (ECP) are offered to users that empower them to safeguard private keys helping to ensure keys remain inaccessible to attackers without physical device access.

CBA allows you to enforce certificate-based authentication for all of Google Cloud, including specific resources (VM, Console, API), individual groups or organizations, across multiple end user vectors from browser to gcloud and terraform command-line-interfaces. It is integrated with many services in Google Cloud for effective access enforcement to cloud resources. For example, [Context-Aware Access](#) for Google APIs and the Cloud Console, [Identity Aware Proxy](#) (IAP) for web apps and workloads, and [VPC Service Controls](#) (VPC-SC) for network enforcement.





Certificate-based access (CBA) architecture overview.

How to implement Certificate-based access

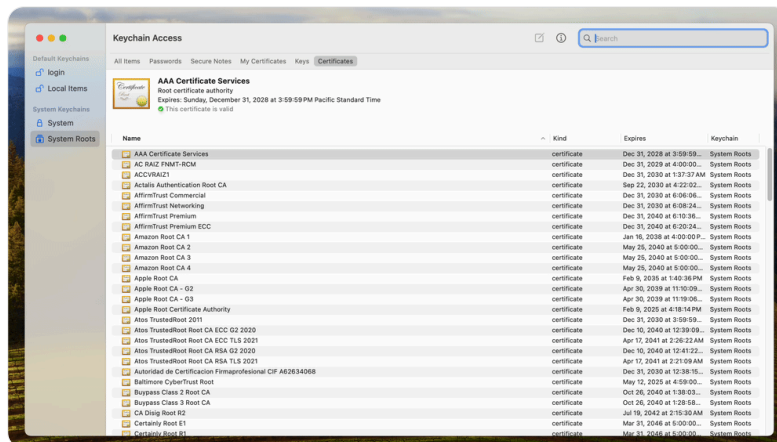
Certificate-based access operates on two core pillars:

- **Access Levels:** These act as the foundation for defining granular access control policies within Google Cloud. Access levels include a set of users and the access requirements necessary; an access level can require Certificates, but users can also define access levels using a wide variety of alternative signals.
- **Certificate Identification:** CBA relies on X.509 certificates as robust device identifiers. Google Cloud offers its own [Certificate Authority Service](#), but most external Certificate Authority service providers are compatible as well.

Step 1: Certificate provisioning

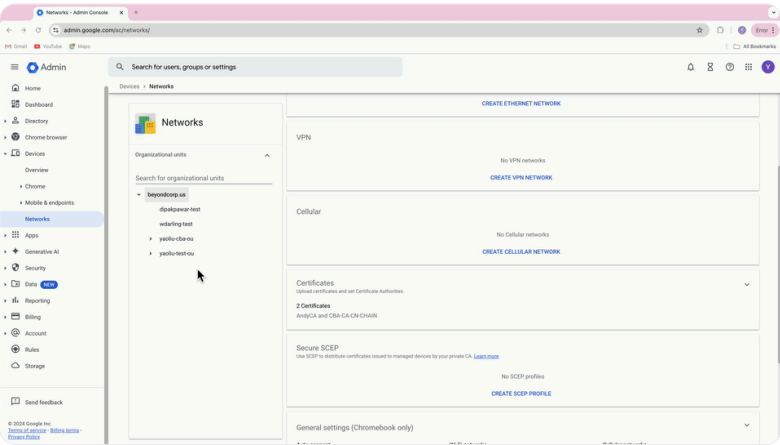
Choose the certificate options that work for you. We support customers' Certificate Authority providers (CA) and existing certificates on the devices. You can use [Endpoint Verification certificates](#) or Google's [Certificate Authority Services](#) for a fully-managed private CA.

We recommend the certificates to be stored securely in OS keystores or TPM and never expose the associated private keys to applications. CBA supports offloading private key operations to these secure storage via [ECP](#) and browser for applications built using [Google Cloud SDK](#) or web.



Step 2: Certificate registration

Administrators can configure an organization policy via the [Google Admin console](#) and instruct their corporate managed Chrome devices to register the chosen certificates to be accepted by CBA policy enforcement and use them in mTLS handshakes. The certificate registration is performed by the [Endpoint Verification Chrome extension](#).

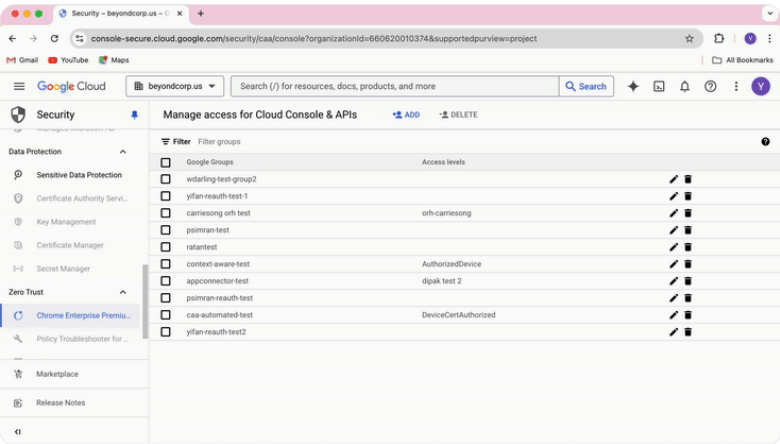


Certificate Registration using Endpoint Verification.

Step 3: Policy definition and enforcement

Last, admins will write CBA policies and enforce mTLS for accessing Google Cloud resources. The policy is implemented depending on the types of resources they want to protect. These policies apply to Google Cloud Console, gcloud CLI, Terraform, and third-party applications.

- [Protecting access to Cloud Console and Google Cloud APIs](#)
- [Protecting access to Web Apps](#)
- [Protecting access to VMs](#)



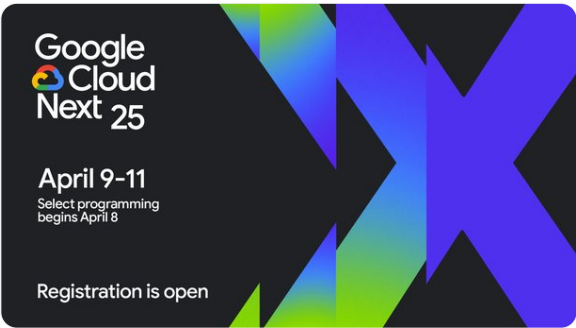
Create CBA Policy Enforcing mTLS

What's next

means of identifying your users. To learn more about implementing CBA as part of your overall security strategy for Google Cloud, please refer to our [documentation](#).

Posted in [Security & Identity](#)

Related articles



Security & Identity

Get ready for a unique, immersive security experience at Next '25

By Robert Sadowski • 5-minute read



Security & Identity

Introducing Google Cloud Abuse Event Logging to enable automated incident remediation

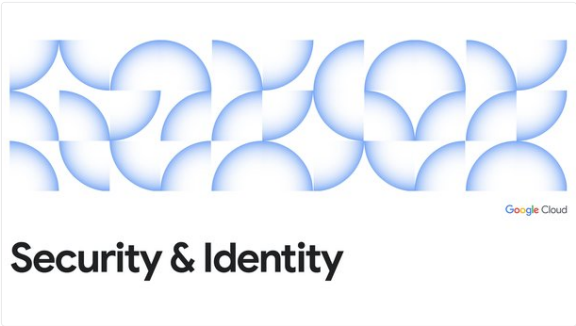
By Alex Dininger • 4-minute read



Security & Identity

Cloud CISO Perspectives: From gen AI to threat intelligence: 2024 in review

By Phil Venables • 5-minute read



Security & Identity

How Google Cloud can help customers achieve compliance with NIS2

By Tara Brady • 9-minute read

Follow us



Google Cloud

Google Cloud Products

Privacy

Terms



Help

English

EXHIBIT 7

Understand mutual TLS at Google Cloud

Mutual TLS (mTLS) is an industry standard protocol for mutual authentication between a client and a server. The mTLS protocol ensures that both the client and server, at each end of a network connection, are who they claim they are by verifying that both possess the private key associated with the client certificate.

What is a client certificate?

A client certificate, also called a Transport Layer Security (TLS) certificate, is a file that contains important information for verifying a device's identity. The certificate information includes the public key, a statement of who issued the certificate (certificates can be issued by certificate authorities or self-signed), and the certificate's expiration date.

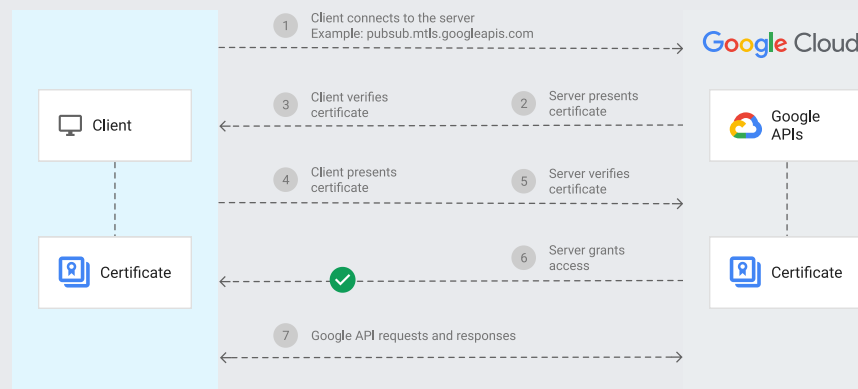
How the Google APIs validate device identity

The TLS protocol uses a technique called public key infrastructure (PKI), which relies on a pair of asymmetric keys: a public key and a private key. Anything encrypted with the private key can be decrypted only with the public key. The Google Cloud APIs use the TLS protocol to verify the identity of a device by decrypting the message encrypted by the private key using the public key of the certificate during the mTLS handshake. The successful decryption proves the possession of the private key which is only available from trusted devices.

To enable the mTLS handshake and validation process, a client must do the following:

- Establish an mTLS connection with the Google APIs by using mTLS-specific API endpoints. The mTLS-specific endpoints have the following format: `[service].mtls.googleapis.com`
- Discover and use the device certificate during the mTLS handshake. If you are using Endpoint Verification for certificate deployment, this type of certificate is automatically discovered and used by the supported clients.

The following diagram illustrates the mTLS handshake between a client and a Google API server:



What's next

- [Set up certificate-based access](https://cloud.google.com/beyondcorp-enterprise/docs/set-up-cba) (/beyondcorp-enterprise/docs/set-up-cba)

1/14/25, 3:09 PM

Understand mutual TLS at Google Cloud | BeyondCorp Enterprise

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-01-08 UTC.

EXHIBIT 8

Deploy Endpoint Verification to use with certificate-based access

As part of a Chrome Enterprise Premium solution, Endpoint Verification provides critical device trust and security-based access control, and can help enforce fine-grained access control on your Google Cloud resources.

For more information about Endpoint Verification, including how to install it, see [Endpoint Verification overview](#) (/endpoint-verification/docs/overview).

If you are installing Endpoint Verification for the first time, we recommend installing the latest version. The following versions of Endpoint Verification are required to use certificate-based access:

- Endpoint Verification Chrome Extension: 1.0.38 or later
- Native helper: 0.4.40.0 or later for Mac, 0.4.36.0 or later for Windows, and 20191007 or later for Linux

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](#) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](#) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](#) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-01-08 UTC.

EXHIBIT 9

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

New Google API Will Securely Verify Chrome Devices



Light Dark

September 19, 2016

By [Larry Loeb](#)

2 min read

<https://securityintelligence.com/news/new-google-api-will-securely-verify-chrome-devices/>



1/7

Ex. 9

K.Mizra LLC, Ex. 2019, p. 140
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

[Google](#) announced a new feature and administration serverside API for the Chrome OS called Verified Access. This API will cryptographically validate the identity of any Chrome OS device that has connected to an enterprise's network. Additionally, it will allow the enterprise to verify that any connected devices conform to the company's security policies.

The new Google API uses digital certificates stored in the hardware-based Trusted Platform Modules (TPMs) that are found in every Chrome OS device. The TPM creates a way to ensure the correct security state of the devices has not been altered.

Normal security policy checks will look for inconsistencies and red flags, but the results of the check may be invalid if the device has been compromised. That's the problem Verified Access is trying to address.

The Magic of Google API

The cloud-based Google Admin console can help enterprises assess networks, servers and intranet pages that use mutual TLS-based authentication through the Verified Access API. The API is designed to allow a network service, such as a VPN gateway or an enterprise certificate authority (CA), to obtain a hardware-backed cryptographic guarantee of the identities of the device and user trying to access it.

The API doesn't do its magic for free, however. A [Google page](#) said the network service endpoint must be configured specifically for the API to be valid. End users will need to install a special extension on their Chrome OS devices to start.

Network services that need to be protected must also understand the Verified Access protocol. Their services must change how they work to allow for the new API. For this reason, Google encouraged [security services](#) to integrate their products with the API.

Through the proper use of the API, enterprises will be able to assure that Chrome OS users have the proper verified boot mode device policy or user policy, since they are specified by the domain's admin.

New Smart Card Authorization

<https://securityintelligence.com/news/new-google-api-will-securely-verify-chrome-devices/>

2/7

Ex. 9

K.Mizra LLC, Ex. 2019, p. 141
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

organizations.

It plays off Citrix Receiver for Chrome 2.1. Users can now authenticate to virtualized Citrix apps using smart cards. They can use single sign-on, login once to their Chromebooks, and then be authenticated across Citrix and virtualized Windows apps.

[Application Programming Interface \(API\)](#) | [Google](#) | [Google Chrome](#) | [Identity](#) | [Identity and Access Management \(IAM\)](#) | [Operating System \(OS\)](#)

Larry Loeb

Principal, PBC
Enterprises

POPULAR



CLOUD SECURITY | January 22, 2025

2024 Cloud Threat Landscape Report: How does cloud security fail?

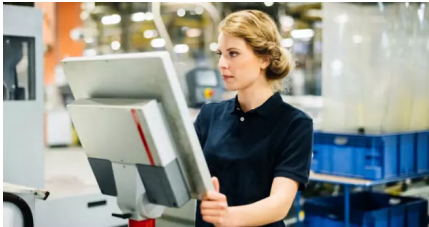
4 min read - Organizations often set up security rules to help reduce cybersecurity vulnerabilities and risks. The 2024 Cost of a Data Breach Report discovered that 40% of all data breaches involved data distributed across multipl...

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices



4 min read - Grocery store shoppers at many chains recently ran into an unwelcome surprise: empty shelves and delayed prescriptions. In early November, Ahold Delhaize USA was the victim of a cyberattack that...



RISK MANAGEMENT | October 25, 2024

Addressing growing concerns about cybersecurity in manufacturing

4 min read - Manufacturing has become increasingly reliant on modern technology, including industrial control systems (ICS), Internet of Things (IoT) devices and operational technology (OT). While these innovations boost productivi...

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

Cost of a Data Breach Report 2024

Read the report →

MORE FROM



January 31, 2025



January 30, 2025

<https://securityintelligence.com/news/new-google-api-will-securely-verify-chrome-devices/>

5/7

Ex. 9

K.Mizra LLC, Ex. 2019, p. 144
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

4 min read - "A computer can never be held accountable, therefore a computer must never make a management decision."- IBM Training Manual, 1979Artificial intelligence (AI) adoptio...

4 min read - As ransomware attacks continue to escalate, their toll is often measured in data loss and financial strain. But what about the loss of human life? Nowhere is the ransomware threat...



January 29, 2025

AI and cloud vulnerabilities aren't the only threats facing CISOs today

6 min read - With cloud infrastructure and, more recently, artificial intelligence (AI) systems becoming prime targets for attackers, security leaders are laser-focused on defending these...

1/31/25, 11:20 AM

New Google API Will Securely Verify Chrome Devices

Topic updates

Get email updates and stay ahead of the latest threats to the security landscape, thought leadership and research.

Subscribe today →

Analysis and insights from hundreds of the brightest minds in the cybersecurity industry to help you prove compliance, grow business and stop threats.

Cybersecurity News

By Topic

Follow us on social

By Industry

Exclusive Series

X-Force

Podcast

Events

Contact

About Us

© 2025 IBM Contact Privacy Terms of use Accessibility Cookie Preferences

Sponsored by

EXHIBIT 10

1/31/25, 11:57 AM

Google Workspace Updates: Context Aware Access insights and recommendations are now generally available

Google Workspace Updates

This official feed from the Google Workspace team provides essential information about new features and improvements for Google Workspace customers.

1/31/25, 11:57 AM

Google Workspace Updates: Context Aware Access insights and recommendations are now generally available

Context Aware Access insights and recommendations are now generally available

Thursday, October 31, 2024

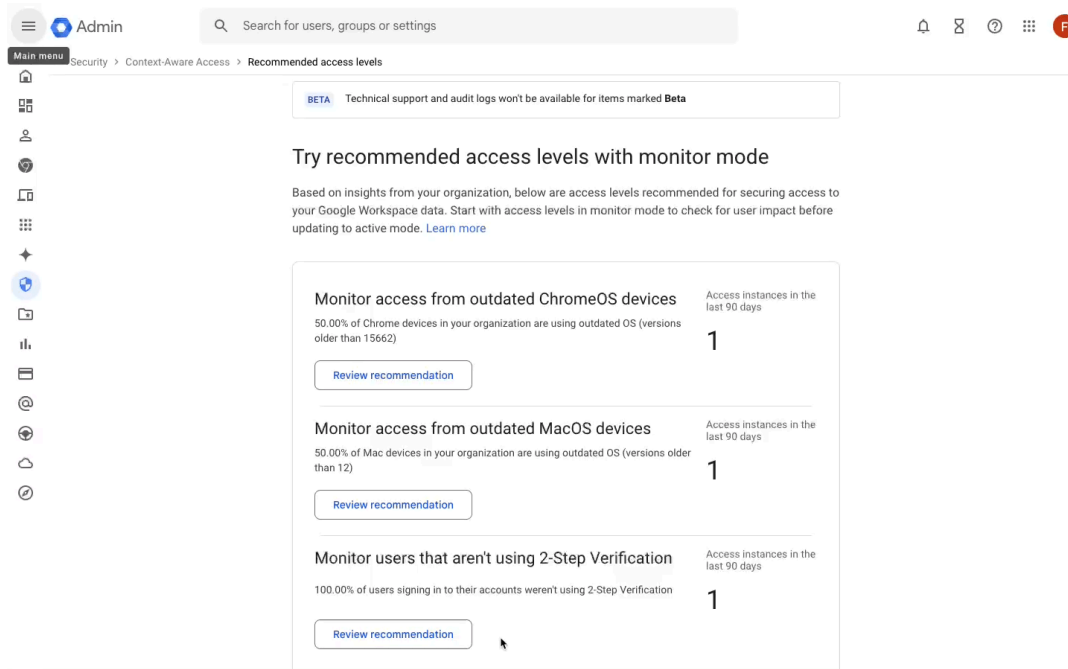
What's changing

We're making it easier to apply [context-aware access \(CAA\)](#) policies with new insights and recommendations. We'll proactively surface potential security gaps and suggest pre-built CAA levels which admins can deploy to remediate the security gaps. These insights will be surfaced to customers, if they have not deployed any CAA policies to their users. When you deploy a recommendation, it will first be placed in [Monitor Mode](#), so you can understand how the policy will block user access over time, and can be reviewed in the [CAA audit logs](#).

With this release, we've also added the ability for admins to customize the recommendations as they see fit before they're applied broadly. Additionally, we'll send primary admins an email on a quarterly basis with insights and actionable recommendations.

1/31/25, 11:57 AM

Google Workspace Updates: Context Aware Access insights and recommendations are now generally available



Who's impacted

Admins

Why it's important

Using Context-Aware Access, admins can set up different access levels based on a user's identity and the context of the request (location, device security status, IP address). This can help provide granular access controls without the need for a VPN, and give users access to Google Workspace resources based on organizational policies. Insights and recommendations help admins improve the cybersecurity posture of their organization by proactively identifying areas that need attention, significantly reducing the need for admins to identify these risks themselves. For example, if we detect devices with outdated operating system versions accessing corporate Workspace data, we can surface this as an Insight & pair it with a recommendation to block such devices from accessing Workspace data with a few clicks.

1/31/25, 11:57 AM

Google Workspace Updates: Context Aware Access insights and recommendations are now generally available

Getting started

- **Admins:** Visit the Help Center to learn more about protecting your business with [Context-Aware Access](#) and [applying recommended access levels](#).
- **End users:** There is no end user impact or action required.

Rollout pace

- [Rapid Release and Scheduled Release domains](#): Available now.

Availability

Available to Google Workspace

- **Enterprise** Standard and Plus
- **Education** Standard and Plus
- **Enterprise** Essentials Plus
- Also available to **Cloud Identity Premium** customers

Resources

- [Google Workspace Admin Help: Protect your business with Context-Aware Access](#)



Labels: [Admin console](#) , [Rapid Release](#) , [Scheduled Release](#) , [Security and Compliance](#)



1/31/25, 11:57 AM

Google Workspace Updates: Context Aware Access insights and recommendations are now generally available



[Google](#) · [Privacy](#) · [Terms](#)

EXHIBIT 11

1/31/25, 12:00 PM

Google Workspace Updates: Block compromised mobile devices using context-aware access

Google Workspace Updates

This official feed from the Google Workspace team provides essential information about new features and improvements for Google Workspace customers.

1/31/25, 12:00 PM

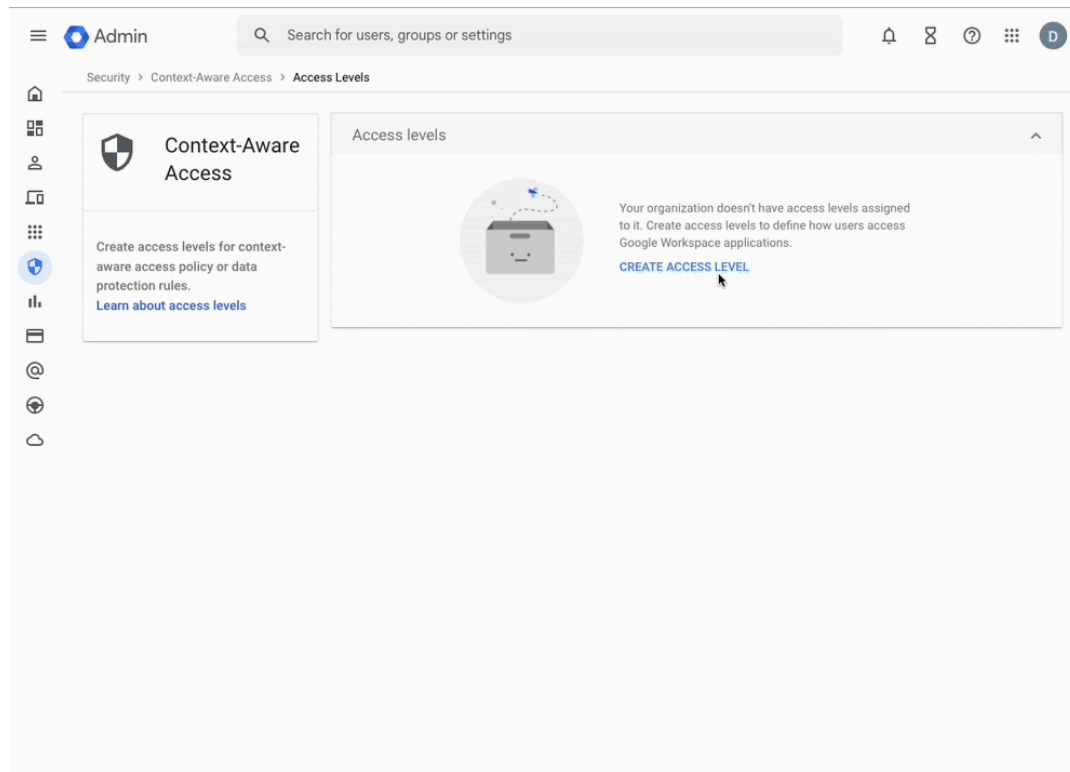
Google Workspace Updates: Block compromised mobile devices using context-aware access

Block compromised mobile devices using context-aware access

Monday, May 6, 2024

What's changing

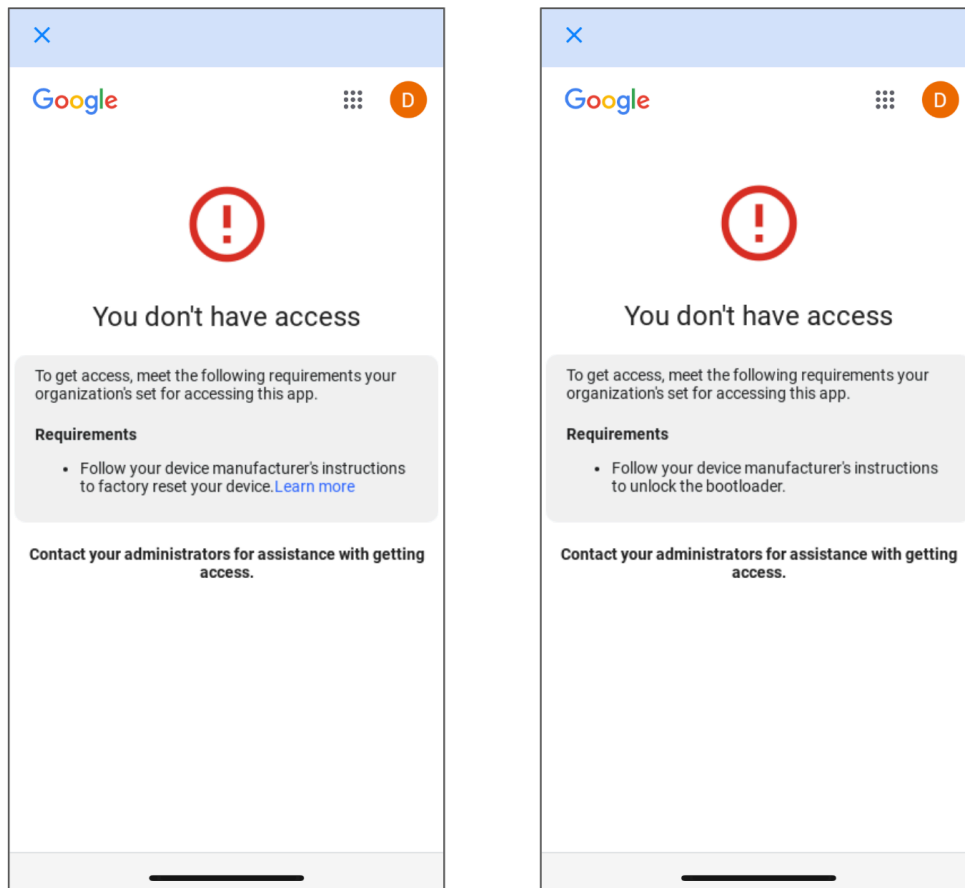
Using context-aware access, you now have the option to automatically block access to Google Workspace data from compromised Android and iOS devices. A device may be counted as compromised if certain unusual events are detected, including devices that are jailbroken, bypassing of security controls, modification of restricted settings, and more.



Creating a new rule to block compromised mobile devices

1/31/25, 12:00 PM

Google Workspace Updates: Block compromised mobile devices using context-aware access



Blocking message for compromised iOS and Android devices

Getting started

- **Admins:** Visit the Help Center to learn more about how to [create context conditions and apply rules with compromised device signals](#), and our Developer documentation regarding [custom access level specification](#). Compromised device events reports can be [viewed in the security center](#) by going to *Security > Security center > dashboard > compromised device events*.

Rollout pace

<https://workspaceupdates.googleblog.com/2024/05/block-compromised-mobile-devices-using-context-aware--access.html>

3/5

Ex. 11

K.Mizra LLC, Ex. 2019, p. 156
 Google LLC v. K.Mizra LLC
 IPR2025-01436

1/31/25, 12:00 PM

Google Workspace Updates: Block compromised mobile devices using context-aware access

- **Block access to Google Workspace data:** available immediately for both Android and iOS.
- **Remediation message:** available immediately for Android, available on May 9, 2024 for iOS.

Availability

Available to Google Workspace

- **Enterprise** Standard and Plus
- **Education** Standard and Plus
- **Frontline** Standard
- **Enterprise Essentials** Plus
- **Cloud Identity** Premium

Resources

- [Google Workspace Admin Help: Set up Context-Aware Access](#)
- [Access Context Manager Help: Custom access level specification](#)
- [Google Workspace Admin Help: Compromised device events report](#)



Labels: [Admin console](#) , [MDM](#) , [Rapid Release](#) , [Scheduled Release](#) , [Security and Compliance](#)



1/31/25, 12:00 PM

Google Workspace Updates: Block compromised mobile devices using context-aware access



[Google](#) · [Privacy](#) · [Terms](#)

EXHIBIT 12

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help

Control access to apps based on user & device context

Allow users to unblock apps with remediation messages in Context Aware Access

Use remediation messages to help users unblock themselves

Next: Use Context-Aware Access with configuration groups

Using remediation messages and custom messages in Context-Aware Access, you can help users unblock themselves when a policy prevents them from accessing an app. These optional (but recommended) messages can help get users back to productivity and reduce support calls for admins.

For example, say that a user on a mobile device is using Gmail in the office successfully during the day, but is blocked when they try to access Gmail at home in the evening. When remediation messages are enabled, they will see guidance on how to address the reason they are blocked.

Remediation and custom messages are supported for access levels created in both Basic mode and Advanced mode. Also, they are supported for both Core Services and SAML apps.

Use remediation messages and custom messages to help your users unblock themselves

When blocked, your users can encounter:

- **Default message**—Displays if you have not added remediation messages or custom messages. An example default message is: **Your organization's policy is blocking access to this app.**
- **Remediation messages**—Replaces the default message. The messages are system generated, and correspond to the specific policy violation that blocked the user. Remediation messages can present several remediation options to the user, which they can expand by clicking **Show more options**. In the case of several remediation options, the user needs to complete the steps for any one of the available options to unblock themselves.
- **Custom message**—Adds specific help for the user, such as additional advice on getting unblocked or a helpful link to click. You add custom messages as needed. A custom message can appear in conjunction with the default message, or with remediation messages.

This table shows the interactivity of these messages:

Remediation messages turned on?	Custom message added?	Messages the user sees
No	No	Default message only
Yes	No	Remediation messages only. In some cases the default message might display if the remediation messages can't be generated.
No	Yes	Default message and custom message
Yes	Yes	Remediation messages and custom message

Understand remediation messages

Each remediation action corresponds to an attribute which is causing access to be denied. The following table summarizes possible remediation messages that the user might see. The messages are

<https://support.google.com/a/answer/11560430?sjid=18176077754207218303-EU>

1/7

Ex. 12

K.Mizra LLC, Ex. 2019, p. 160
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help

created systematically depending on the policy that was violated.

Note that different messages can be shown for the same attribute according to the expectation in the access level. For example, if the access level is `device.screen_lock_enabled == true`, the message is **Set a screen password on your device**. If access level is `device.screen_lock_enabled == false`, the message is **Remove the screen password from your device**. Removing a screen password could be less secure, so the user should confirm this action with the admin.

Actual messages might differ from the messages displayed below.

Attribute	Message
Admin approval	Switch to a device approved by your organization. Contact your admin if you don't have access to one.
	Switch to a device that's not associated with your organization. Note, this could be less secure, so you may want to confirm this with your admin.
Company owned device	Switch to a device owned by your organization. Contact your admin if you don't have access to one.
	Switch to a device that's not owned by your organization.
CTS profile match	Reset your device to factory settings.
	Your device can't access this app with an OEM Android installation. Contact your admin for more info.
Encryption	Switch to a device that has one of the following encryption statuses: [status1, status2].
	Switch to a device that doesn't have the following encryption status: [status]
Has potentially harmful apps	Uninstall any apps listed by Google Play Protect as potentially harmful.
	Your device can't access this app with currently installed apps. Contact your admin for more info.
IP address	You can't access this app from your current IP subnetwork. Contact your admin to learn more.
OS type	Switch to a device that uses one of the following: [os1, os2]
	Switch to a device that doesn't use: os1
OS version	Update your device to [OS version X] or higher
	Update your device to an OS version lower than [OS version X]
Partner Attributes	Install [PARTNER NAME] on your device. ¹
	Your device isn't meeting some requirements, based on information from [PARTNER NAME]. ²
Region code	You can't access this app from your current location. Contact your admin to learn more.
Screen lock	Set a screen password on your device.
	Remove the screen password from your device. Note, this could be less secure, so you may want to confirm this with your admin.
Verify apps	Enable Google Play Protect on your device.
	Disable Google Play Protect on your device.
Verified boot	Follow your device manufacturer's instructions to lock the bootloader.
	Follow your device manufacturer's instructions to unlock the bootloader.
Verified Chrome OS	Install a verified Chrome OS on your device.
	You can't access this app with a verified Chrome OS.

¹ Make sure the partner security app (for example, Lookout) is installed on the device. If it's installed but the user is still seeing this message, the device might not be properly enrolled into the partner MDM. Check the partner dashboard to verify if this is the case. If required, contact the partner to resolve the issue.

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help

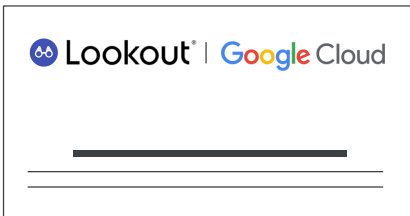
² Check the partner app on the device for more details. If required, contact the partner to resolve the issue.

Understand remediation messages and third-party partner integrations

As an administrator, you can integrate supported third-party partners (those that are part of the [BeyondCorp Alliance](#)) with Google endpoint management in Google Admin console. This informational text displays in the remediation message interface to explain that partner messages can be available to users:



For example, from Lookout:



For details, go to [Set up third-party partner integrations](#).

Common errors to resolve before remediation or custom messages can be seen by the user

These errors must be cleared before users can view remediation messages:

Your device can't be recognized. There may be different steps depending on your device type.

Google doesn't recognize the login device. The remediation step depends on the platform.

- **Desktop devices**—Users must use a Chrome profile with the Endpoint Verification extension installed. Users can't login to Google Workspace apps through incognito, guest, or personal profiles. Note that this error message can display when a user tries to sign in to a new device for the first time. In that case, the user must [sync with the Endpoint Verification extension](#) and refresh the browser.
- **Mobile devices**—For devices to be recognized by CAA, they need to be managed by Google endpoint management (basic or advanced management). For details, go to [Manage devices with Google endpoint management](#). Additionally, devices might need to sync before Google can recognize where the login occurred. For details, see [Sync your device](#).

Sync your device

- **Desktop devices**—Users must [sync with the Endpoint Verification extension](#).
- **Desktop devices**—Devices under advanced management can sync with the device policy app. Devices under basic management can wait for the device to sync automatically, or the user can re-login into any Google app. Let the user know that device sync can take some time.
- **iOS devices**—Google sessions across different applications are tracked with sessions or tokens in Safari. If the Safari tokens are deleted (manually by the user or [Apple ITP](#)), the subsequent logins cannot be mapped to the original logged in device. This can cause the new logins to be blocked with "Your device can't be recognized. There may be different steps depending on your device type" message if remediation is enabled. This issue can occur both on basic and advanced managed devices.

The user needs to complete the following steps to unblock themselves:

<https://support.google.com/a/answer/11560430?sjid=18176077754207218303-EU>

3/7

Ex. 12

K.Mizra LLC, Ex. 2019, p. 162
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help

1. Remove the Google enterprise account from all Google applications.
Log out the Google account from Safari and remove it from any non-Google application which might be using it.
2. [Delete Safari cookies and cache](#) .
3. Log into any Google first-party application, such as Drive or Gmail.
4. Access all of the remaining Google first-party applications to verify that you have access.
5. If access to non-Google applications is needed, sign in to the applications within a few days of logging into the Google applications.
If you do not sign in within 30 days of step 3 and the application is blocked, restart from step 1.

Implement remediation or custom messages

Turn on remediation messages

1. [Sign in](#) with an *administrator* account to the [Google Admin console](#).
If you aren't using an administrator account, you can't access the Admin console.
2. From the Admin console Home page, go to **Security** > Access and data control > **Context-Aware Access**.
3. Select **User message**.
4. Under Remediation messages, click **OFF** and slide right to turn the messages **ON**. You'll see a check mark.
5. Click **Save**.
You can also add a custom message at this point.

Add a custom message

1. [Sign in](#) with an *administrator* account to the [Google Admin console](#).
If you aren't using an administrator account, you can't access the Admin console.
2. From the Admin console Home page, go to **Security** > **Access and data control** > **Context-Aware Access**.
3. Select **User message**.
4. Under Additional custom message, enter your message.
5. Click **Preview** to see what the user will see.
6. Click **Save**.

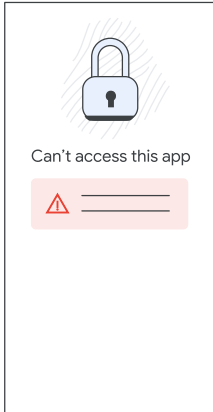
User experience for remediation and custom messages

Default message only

This is an example of a message the user sees if no remediation messages or custom message is configured.

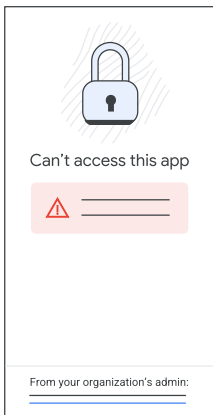
1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help



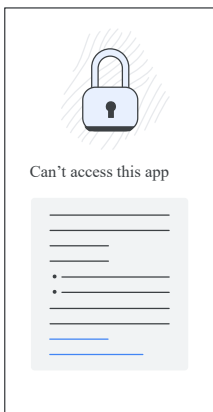
Default message and custom message

This is an example of a message the user sees if no remediation messages are configured, but the custom message is provided.



Remediation message only

This is an example of a message the user sees if remediation messages are configured with no custom message. The user clicks **Show more options** to expand the remediation steps.



Remediation and custom message

This is an example of a message the user sees if both remediation messages and the custom message are configured. The user clicks **Show more options** to expand the remediation steps.

<https://support.google.com/a/answer/11560430?sjid=18176077754207218303-EU>

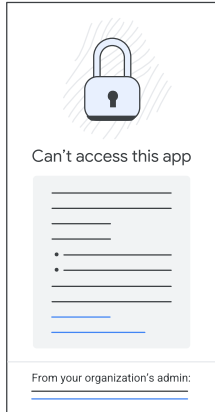
5/7

Ex. 12

K.Mizra LLC, Ex. 2019, p. 164
Google LLC v. K.Mizra LLC
IPR2025-01436

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help



Context-Aware Access remediation and custom messages FAQ

[Expand all](#) | [Collapse all](#)

[Can remediation cause any additional Access Denied cases?](#)

[Why don't the remediation messages reflect the current policies?](#)

[How long does it take for the user to get access after completing the remediation actions?](#)

[Why do Users still see the same remediation options after completing the remediation action?](#)

[Why do remediation message options change without any action on the device?](#)

[Why are remediation messages missing even though they are enabled?](#)

[Do users see the custom user message if remediation is enabled?](#)

[How do I enable remediations for Device policies?](#)

[Why does the **Your device can't be recognized** remediation message display if the Endpoint Verification extension is syncing?](#)

Google, Google Workspace, and related marks and logos are trademarks of Google LLC. All other company and product names are trademarks of the companies with which they are associated.

[Next: Use Context-Aware Access with configuration groups](#)

Need more help?

Try these next steps:



Post to the help community

Get answers from community members

1/31/25, 12:01 PM

Allow users to unblock apps with remediation messages in Context Aware Access - Google Workspace Admin Help



Contact us

Tell us more and we'll help you get there

EXHIBIT 13

Remediate denied access with the Policy Remediator

Preview – Policy Remediator

This feature is subject to the "Pre-GA Offerings Terms" in the General Service Terms section of the [Service Specific Terms](#) (/terms/service-terms#1). Pre-GA features are available "as is" and might have limited support. For more information, see the [launch stage descriptions](#) (/products#product-launch-stages).

This page shows you how to enable and use the Chrome Enterprise Premium Policy Remediator.

Save page

When users attempt to access a Google Cloud resource but aren't compliant with the access policy for the resource, they are denied access and receive a general 403 error message. You can use the Policy Remediator to provide users with actionable steps that they can take to remediate their issue before reaching out to an admin for additional help. The specific remediation actions depend on the access policies, but can include things such as enabling screen lock, updating the operating system (OS) version, or accessing an app from a network allowed by your company.

Enable Policy Remediator

1. Grant your organization admin the `roles/policyremediatormanager.policyRemediatorAdmin` role at the organization level by running the following commands in the Google Cloud CLI:

```
gcloud organizations add-iam-policy-binding 'organizations/ORGANIZATION_ID'
--member PRINCIPAL \
--role roles/policyremediatormanager.policyRemediatorAdmin
```

Replace the following:

- **ORGANIZATION_ID**: the Google Cloud organization ID.

- **PRINCIPAL**: the identifier for the principal, or member, which usually has the following form: `PRINCIPAL_TYPE:ID`. For example, `user:my-user@example.com`.

2. Enable the Policy Remediator Manager API by running the following command:

```
gcloud services enable policyremediatorservice.googleapis.com
```

3. Call the Policy Remediator Manager to enable Policy Remediator for the projects in an organization, this creates a service agent (`/iam/docs/service-agents`).

```
curl -X POST \
  "https://policyremediatorservice.googleapis.com/v1alpha/organizations/ORGANIZATION_ID" \
  --header 'Authorization: Bearer ACCESS_TOKEN' \
  --header 'X-Goog-User-Project: PROJECT_ID'
```

Save page

Replace the following:

- **ORGANIZATION_ID**: the Google Cloud organization ID.
- **ACCESS_TOKEN**: use the following command to generate the access token.

```
gcloud auth print-access-token
```

- **PROJECT_ID**: the Google Cloud project ID.

Following is an example response, which contains the service agent details:

```
{
  "name": "organizations/ORGANIZATION_ID/locations/global/operations/",
  "metadata": {
    "@type":
      "type.googleapis.com/google.cloud.policyremediatorservice.
      a.OperationMetadata",
    "createTime": "",
  }
}
```

```

    "target": "organizations/ORGANIZATION_ID/locations/global/remediatorSe
    "verb": "update",
    "requestedCancellation": false,
    "apiVersion": "v1alpha"
  },
  "done": false
}

```

Where **ORGANIZATION_ID** is the Google Cloud organization ID.

4. In the Google Cloud CLI, run the following command to access the service agent that you created:

Save page

```

curl -X GET \
  "https://policyremediatormanager.googleapis.com/v1alpha/organizations/ORGAN.
  --header 'Authorization: Bearer ACCESS_TOKEN' \
  --header 'X-Goog-User-Project:PROJECT_ID'

```

Replace the following:

- **ORGANIZATION_ID**: the Google Cloud organization ID.
- **ACCESS_TOKEN**: use the following command to generate the access token.

```
gcloud auth print-access-token
```

- **PROJECT_ID**: the Google Cloud project ID.

You should receive the service agent email in the following format:

```

{
  "name": "organizations/ORGANIZATION_ID/locations/global/remediatorService
  "state": "ENABLED",

```

```
"serviceAccountEmail": "service-org-ORGANIZATION_ID @gcp-sa-v1-remediator.
}
```

Where *ORGANIZATION_ID* is the Google Cloud organization ID.

Assign the service agent role in the Google Admin console

1. Log in to the Google Admin console.

Go to the Google Admin console (<https://admin.google.com/ac/devices/list>)

2. Go to **Account > Admin roles**, and then click **Create new role**.

Save page

- Enter a name and a description (optional) for the role, and then click **Continue**.
- In **Admin console privileges**, go to **Services > Mobile and Device Management** and select the **Manages Devices and Settings** permission.
- In **Admin API privileges**, go to **Groups**, and then select the **Read** permission.
- Click **Continue**, confirm your entries, and complete creating the role.
- Go to **Assign Service Accounts** and enter the email address of the newly created service agent (`#service-acct-email`).
- Click **Add > Assign Role**.

3. In the Google Cloud CLI, run the following commands to grant the Service Agent (`policyremediator.serviceAgent`) role to the service agent at the organization level. This gives the service agent permission to read the Identity and Access Management and other access policies for your organization.

```
gcloud organizations add-iam-policy-binding 'organizations/' \
  --member='serviceAccount:service-org-ORGANIZATION_ID @gcp-sa-v1-remedia
  --role='roles/policyremediator.serviceAgent'
```

Replace *ORGANIZATION_ID* with the Google Cloud organization ID.

Enable Policy Remediator for an IAP resource

You must have a Chrome Enterprise Premium license to use this feature.

1. Go to the Identity-Aware Proxy (IAP) page.
[Go to IAP](https://console.cloud.google.com/security/iap) (https://console.cloud.google.com/security/iap)
2. Select a resource, and then click **Settings**.
3. Go to **Remediate access**, and then select **Generate remediation actions**.

Grant the remediator role

Save page

To give users permission to remediate denied access to IAP resources, run the following command in the Google Cloud CLI:

```
gcloud iap web add-iam-policy-binding \
  --member=' PRINCIPAL ' \
  --role=' roles/iap.remediatorUser '
```

Replace *PRINCIPAL* with an identifier for the principal, or member, which usually has the following form: `PRINCIPAL_TYPE:ID`. For example, `user:my-user@example.com`.

For additional information, see [gcloud iap web add-iam-policy-binding](/sdk/gcloud/reference/iap/web/add-iam-policy-binding) (/sdk/gcloud/reference/iap/web/add-iam-policy-binding).

To give users permission to remediate access to IAP resources at a project level, run the following command in the Google Cloud CLI:

```
gcloud projects add-iam-policy-binding PROJECT_ID \
  --member PRINCIPAL \
  --role roles/iap.remediatorUser
```

Replace the following:

- **PROJECT_ID**: the Google Cloud project ID.
- **PRINCIPAL**: the identifier for the principal, or member, which usually has the following form: **PRINCIPAL_TYPE:ID**. For example, **user:my-user@example.com**.

Remediate with the Help Desk

When end users are denied access, they are redirected to a Chrome Enterprise Premium page that contains troubleshooting information, including a troubleshooting URL and a remediation token. If users don't have permission to open the remediation token, they can copy the remediation token and send it to the Help Desk for additional help.

[Save page](#)

Policy attributes and associated messages

The following table provides the list of attributes that are supported by the Policy Remediator.

Attribute	Default message
ip_address	You're accessing the app from a network not allowed by your company.
region_code	Access this app from a region allowed by your company.
is_secured_with_screenlock	Set a screen password on your device. Turn off the screen password on your device.
verified_chrome_os	Use a device with verified [OS type]. Use a device without verified [OS type].
is_admin_approved_device	Use a device approved by your organization administrator. Use a device not approved by your organization administrator.
is_corp_owned_device	Use a device owned by your organization. Use a device not owned by your organization.
encryption_status	Use an encrypted device. Use an unencrypted device.

Attribute	Default message
os_type	Switch to a [OS type] device. [OS type] devices cannot access this app.
os_version	Update to an OS version that is at least [version]. Downgrade your OS to a version less than [version].

Troubleshooting

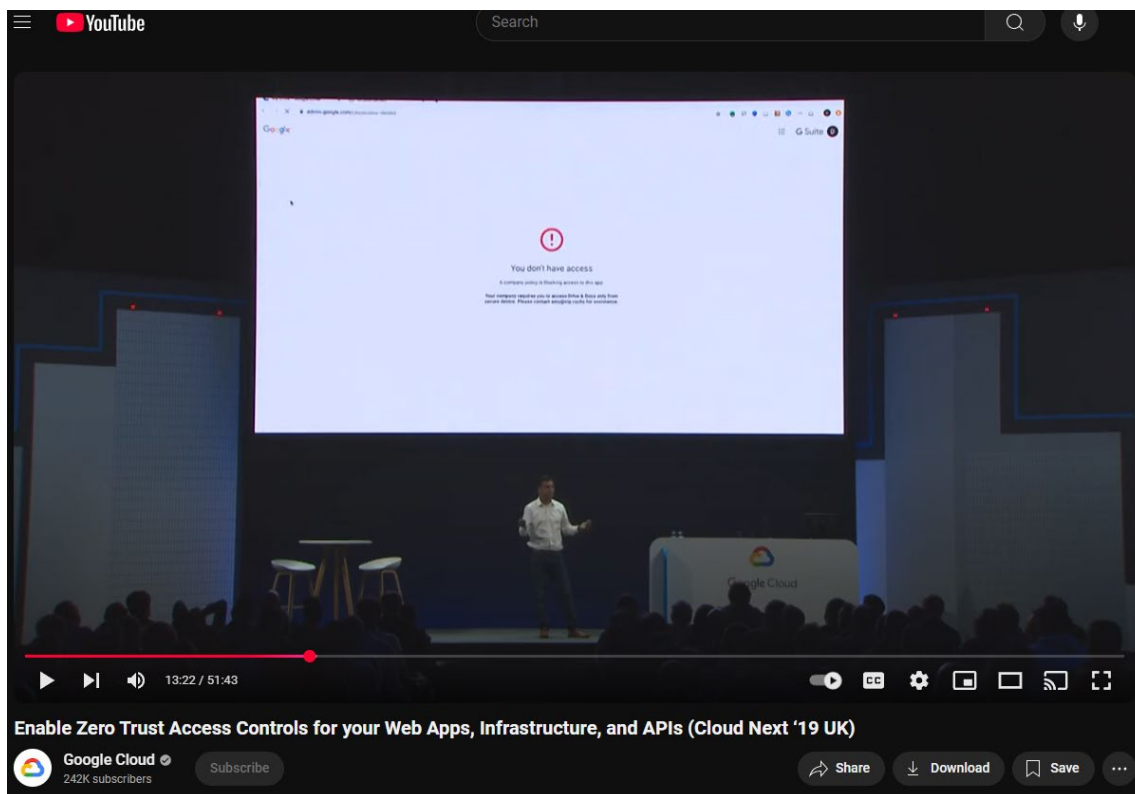
The Policy Remediator cannot generate remediations when any of the following occur:

- A resource has conflicting policies, such as a user must connect using Windows and macOS. Save page
- The attribute is not supported by the Policy Remediator.
- The service agent does not have permission to remediate.

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (<https://creativecommons.org/licenses/by/4.0/>), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (<https://www.apache.org/licenses/LICENSE-2.0>). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (<https://developers.google.com/site-policies>). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-01-30 UTC.

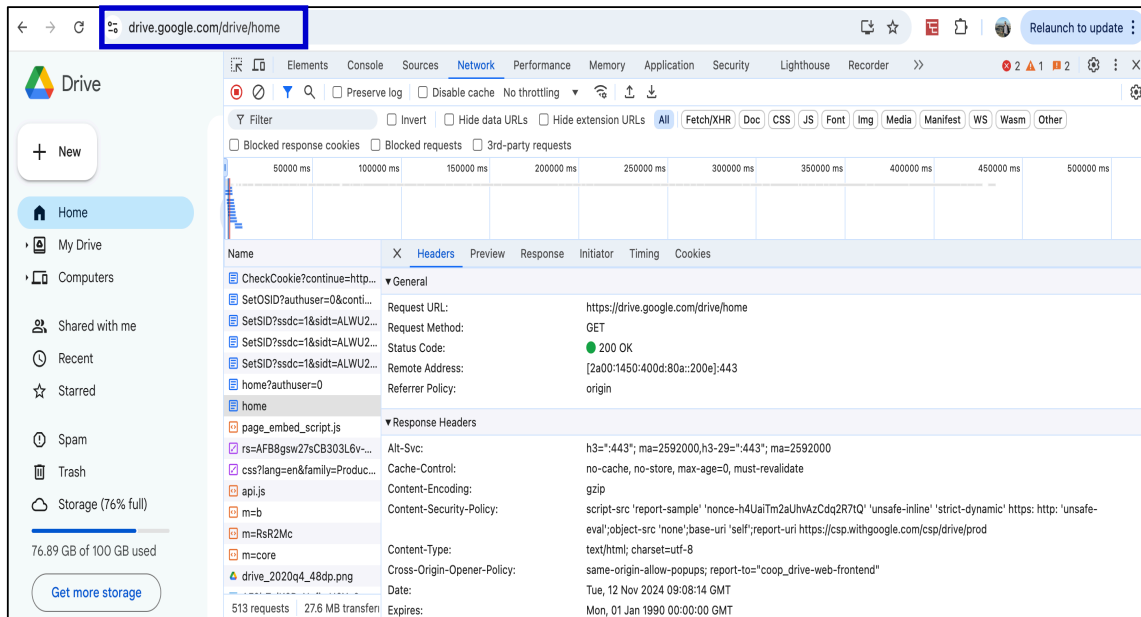
EXHIBIT 14



<https://www.youtube.com/watch?v=2jz5FQpGWQ4>

Ex. 14, p. 1
K.Mizra LLC, Ex. 2019, p. 176
Google LLC v. K.Mizra LLC
IPR2025-01436

EXHIBIT 15



<https://drive.google.com/drive/home>

Ex. 15, p. 1
 K.Mizra LLC, Ex. 2019, p. 178
 Google LLC v. K.Mizra LLC
 IPR2025-01436

EXHIBIT 16

The screenshot shows a web browser window with the address bar displaying `admin.google.com/caa/access-denied?pli=1`. The main content area shows a red exclamation mark icon and the text "You don't have access". Below this, a message states: "Your organization's policy is blocking access to this app. Contact your administrators for assistance with getting access." The browser's developer tools are open, showing the Network tab. A list of requests is visible, with the first request, `access-denied?pli=1`, selected. The details for this request are shown on the right, including the Request URL, Request Method (GET), Status Code (200 OK), Remote Address, and Referrer Policy. The Response Headers section is also expanded, showing various security-related headers.

Access Denied Message:

You don't have access

Your organization's policy is blocking access to this app.
Contact your administrators for assistance with getting access.

Network Details:

Name	Request URL	Request Method	Status Code	Remote Address	Referrer Policy
access-denied?pli=1	<code>https://admin.google.com/caa/access-denied?pli=1</code>	GET	200 OK	<code>[2a00:1450:400d:802::200e]:443</code>	strict-origin-when-cross-origin

Response Headers:

Header	Value
Accept-Ch	Sec-CH-UA-Arch, Sec-CH-UA-Bitness, Sec-CH-UA-Full-Version, Sec-CH-UA-Full-Version-List, Sec-CH-UA-Model, Sec-CH-UA-WoW64, Sec-CH-UA-Form-Factors, Sec-CH-UA-Platform, Sec-CH-UA-Platform-Version
Alt-Svc	<code>h3=":443"; ma=2592000, h3-29=":443"; ma=2592000</code>
Cache-Control	no-cache, no-store, max-age=0, must-revalidate
Content-Encoding	gzip
Content-Security-Policy	<code>script-src 'report-sample' 'nonce-xYkGQgQfswvj2w4wG7hP4Q' 'unsafe-inline'; object-src 'none'; base-uri 'self'; report-uri /_/_IdentityContextAwareAccessAccessDeniedUI/cspreport; worker-src 'self'</code>