(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
3 August 2017 (03.08.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/131626 A1

(51) International Patent Classification:
G06F 21/56 (2013.01)

(21) International Application Number:
PCT/US2016/014841

(22) International Filing Date:
26 January 2016 (26.01.2016)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **ARUBA NETWORKS, INC.** [US/US]; 1344 Crossman Avenue, Sunnyvale, California 94089 (US).

(72) Inventor: **ARDELI, Ramesh**; 1322 Cross Avenue, Sunnvale, California 94087 (US).

(74) Agents: **FOUGERE, Jeffrey** et al.; Hewlett Packard Enterprise, Mail Stop 79, 3404 E. Harmony Road, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

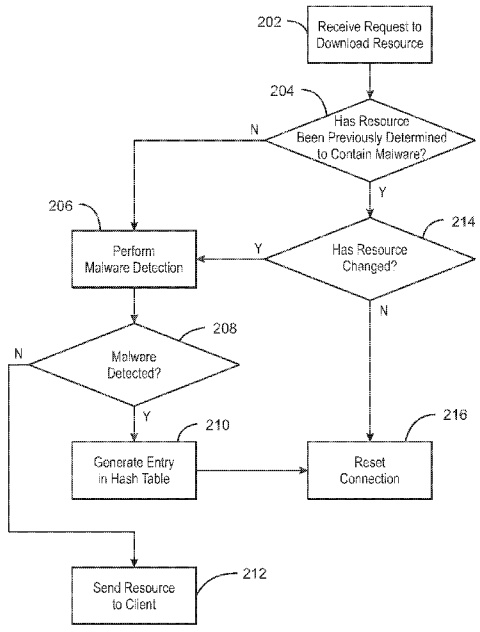
— *as to the identity of the inventor (Rule 4.17(i))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: MALWARE DETECTION

(57) Abstract: An example implementation of the present techniques determines, in response to a request to download a resource, whether the resource has previously been determined to comprise malware. Additionally, it is determined, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination. Further the request to download the resource is terminated if the resource has not changed.



200
FIG. 2

WO 2017/131626 A1

Published:

— *with international search report (Art. 21(3))*

MALWARE DETECTION

BACKGROUND

[0001] Malware refers to a variety of dangerous or otherwise undesirable software that includes viruses, ransomware, spyware, and other, malicious applications. Malware can take the form of executables, scripts, or other infected software that may be downloaded to, and installed on, a computer. In many cases, firewalls are used to detect malware and prevent its installation before it can be do any harm. However, many malware detection approaches are processor intensive, which is computationally expensive, impeding the ability of firewalls to efficiently prevent the spread of malware.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] Certain example implementations are described in the following detailed description and in reference to the drawings, in which:

[0003] Fig. 1 is a block diagram of an example system for detecting malware;

[0004] Fig. 2 is a process flow diagram showing an example method of detecting malware;

[0005] Fig. 3 is a process flow diagram showing an example method of detecting malware;

[0006] Fig. 4 is a block diagram of an example system for detecting malware;

[0007] Fig. 5 is a block diagram of an example system for detecting malware;

[0008] Fig. 6 is a block diagram showing an example non-transitory, tangible computer-readable medium that stores code for detecting malware; and

[0009] Fig. 7 is a block diagram showing an example non-transitory, tangible computer-readable medium that stores code for detecting malware.

DETAILED DESCRIPTION

[0010] As described above, malware detection can be processor intensive. Yet, once malware has been detected, preventing future installation of the detected malware is typically accomplished by performing the same processor intensive operations.

[0011] However, in examples of the present techniques, downloads of previously detected malware are prevented. These downloads are prevented by maintaining a record of detected malware. When resources are requested for download, a check is performed to determine if the resource has been previously determined to be infected with malware. Additionally, a check is performed to see if the resource has changed since the malware was detected. If the resource has changed, the download is allowed to proceed, and traditional malware detection is performed to ensure the changed resource is not infected. If the resource has not changed, the download is prevented. In this way, the firewall detects the malware in the requested resource without downloading the resource and scanning it, thus improving the speed of malware detection.

[0012] Fig. 1 is a block diagram of an example system 100 for detecting malware. The example system is generally referred to by the reference number 100 and can be implemented using the example computing device 602 of Fig. 6 below.

[0013] The example system 100 of Fig. 1 includes clients 102 and a firewall device 104 in communication over a network 106. The system 100 also includes servers 108 in communication with the firewall device 104 over network 110. The clients 102 may be computing devices, such as desktop computers, laptop computers, tablet computers, smartphones, and the like. The firewall device 104 is a security system that monitors and controls the network traffic incoming to, and outgoing from, the network 106. The firewall device 104 includes a unified threat manager (UTM) 112. The UTM 112 is security software that may be capable of performing network firewalling, network intrusion prevention, anti-virus and anti-spam functions, and the like. The network 106 may be a computer communication network, such as a local area network. The servers 108 may be web servers, or ftp servers, that host resources requested by the clients 102. Resources may be files or other content, such as downloadable files and web pages. The network 110 may be a computer communication network or collection of networks, such as the Internet.

[0014] The clients 102 make requests through the firewall device 104 for resources from the servers 108. The firewall device 104 receives the requests from the clients 102, and passes the requests to the servers 108. The servers 108 respond to the requests by downloading the requested resources back to the firewall device 104. The UTM 112 then performs traditional methods to determine if the requested resource is infected with malware. Traditional methods include signature scanning and hash lookup. Signature scanning involves scanning each packet of the requested resource, and comparing the packets to a database of known malware signatures. Hash lookup involves comparing a hash value for the entire resource against a table consisting of hash values for known malware files. Traditional methods may additionally include executing the requested resource in a sandbox environment. A sandbox environment is an execution environment that is isolated to prevent malware from making potentially damaging changes to the firewall device 104 or client 102. Resources that are potentially infected are run in the sandbox environment and monitored to determine if they contain malware. The firewall device 104 may perform these techniques concurrently on multiple requests from multiple clients 102. Typically, using these methods consumes numerous CPU cycles of the firewall device 104, reducing the number of concurrent scans able to be supported by the firewall device 104. However, in implementations of the claimed subject matter, once an infected resource has been detected, these techniques can be avoided, thus increasing the number of concurrent requests that can be scanned.

[0015] If no malware is detected, the resource is determined to be clean, and sent to the requesting client 102. However, if malware is detected, the resource is determined to be infected, the connection between the client 102 and the server 108 is reset, and all data packets received from the server 108 providing the infected resource are dropped. Additionally, an entry for the infected resource is stored locally. Subsequently, when a client 102 requests a resource to be downloaded, the UTM 112 performs a lookup in the local store to determine if the requested resource has been previously determined to be infected with malware. If the lookup is successful, a check is performed to

determine if the requested resource has changed since the malware was detected. If the resource has not changed, the resource is considered still infected. Accordingly, the UTM 112 resets the connection to the server 108 and drops all received packets. If the resource has changed, the entry for the resource is removed from the local store, the resource is downloaded, and the UTM 112 performs the traditional methods to detect malware.

[0016] Fig. 2 is a process flow diagram showing an example method 200 of detecting malware. The example method is generally referred to by the reference number 200 and can be implemented using the processor 608 of the example system 600 of Fig. 6 below.

[0017] The method 200 begins at block 202, where the firewall device 104 receives a request from a client 102 to download a resource. The firewall device 104 passes the request to the server 108 specified in the request, and downloads the resource to the firewall device 104.

[0018] At block 204, the UTM 112 determines whether the resource has been previously determined to contain malware. In an example of the claimed subject matter, a hash lookup table contains an entry for every resource previously determined to contain malware. Each entry consists of a hash value for the absolute uniform resource locator (URL), and the Etag. The Etag is an opaque identifier assigned to the resource by a web server of the server 108. Whenever there is a change to a resource, the web server generates a new Etag for the resource. The Etag is part of the hypertext transfer protocol (HTTP), which is used for cache validation, and instructing a web browser to make a conditional request. With ETag, the browser determines whether to serve the resource locally or from server, if the resource is cached. Since the file is not completely sent to the client, when malware is detected, the client will not be able to cache the resource. The Etag is included in the header packet of the downloaded resource. The header packet is the first packet sent to the firewall device 104 when the resource is downloaded. The lookup is performed using the absolute URL of the requested resource. If the lookup is not successful, i.e., there is no entry in the hash table for the absolute URL, then the

resource has not been previously determined to contain malware. Accordingly, the method flows to block 206.

[0019] At block 206, malware detection is performed using traditional methods. The UTM 112 copies the downloaded resource to local memory of the firewall device, performs signature scanning, hash lookup, and execution in the sandbox environment.

[0020] At block 208, it is determined whether malware is detected using the traditional methods. If malware is detected, the method 200 flows to block 210.

[0021] At block 210, an entry is generated in the hash table for the resource. As stated previously, the entry contains a hash value for the absolute URL of the resource, and the Etag.

[0022] Referring back to block 208, if malware is not detected, the method 200 flows to block 212. At block 212, the downloaded resource is sent to the client 102.

[0023] Referring back to block 204, if the resource has not been previously determined to contain malware, the method flows to block 214. At block 214, the UTM 112 determines whether the resource has changed since malware was detected in this resource. In examples of the claimed subject matter, the UTM 112 compares the Etag in the hash lookup entry to the Etag in the header packet of the resource. If the Etags are the same, there has been no change. Thus, the resource is determined to still contain malware. Accordingly, at block 216, the connection to the server 108 hosting the resource is reset. Additionally, all data packets received over the reset connection are dropped. Dropping the received data packets means the data packets are deleted.

[0024] Referring back to block 214, if the Etags are not the same, there has been a change to the resource. As such, it is possible that the resource no longer contains malware. Accordingly, the entry for the resource is removed from the hash lookup table. Additionally, the method flows to block 206, where malware detection is performed using the traditional methods.

[0025] This process flow diagram is not intended to indicate that the blocks of the example method 200 are to be executed in any particular order, or that all of the blocks are to be included in every case. Further, any number of

additional blocks not shown may be included within the example method 200, depending on the details of the specific implementation.

[0026] Fig. 3 is a process flow diagram showing an example method 300 of detecting malware. The example method is generally referred to by the reference number 300 and can be implemented using the processor 608 of the example system 600 of Fig. 6 below. The method 300 is performed by the UTM 112, and begins at block 302, where, in response to a request to download a resource, the UTM 112 determines whether the resource has previously been determined to comprise malware. As stated previously, the UTM 112 performs a hash lookup based on the absolute URL of the resource. If the lookup is successful, the resource is determined to comprise malware.

[0027] At block 304, if the resource has previously been determined to comprise malware, the UTM 112 determines whether the resource has changed since the previous determination. As stated previously, the UTM 112 compares the Etag in the hash lookup table with the current Etag of the resource. If the two Etags match, the resource is determined to have not changed.

[0028] At block 306, if the resource has not changed, the UTM 112 terminates the request to download the resource.

[0029] Fig. 4 is a block diagram of an example system for detecting malware. The system is generally referred to by the reference number 400.

[0030] The system 400 may include a firewall device 402, and one or more client computers 404, in communication over a network 406. As used herein, the firewall device 402 may include a server, a personal computer, a tablet computer, and the like. As illustrated in Fig. 4, the firewall device 402 may include one or more processors 408, which may be connected through a bus 410 to a display 412, a keyboard 414, one or more input devices 416, and an output device, such as a printer 418. The input devices 416 may include devices such as a mouse or touch screen. The processors 408 may include a single core, multiples cores, or a cluster of cores in a cloud computing architecture. The computing device 402 may also be connected through the bus 410 to a network interface card (NIC) 420. The NIC 420 may connect the firewall device 402 to the network 406.

[0031] The network 406 may be a local area network (LAN), a wide area network (WAN), or another network configuration. The network 406 may include routers, switches, modems, or any other kind of interface device used for interconnection. The network 406 may connect to several client computers 404. Through the network 406, several client computers 404 may connect to the firewall device 402. Further, the firewall device 402 may prevent malware from entering the network 406. The client computers 404 may be similarly structured as the firewall device 402.

[0032] The firewall device 402 may have other units operatively coupled to the processor 408 through the bus 410. These units may include non-transitory, tangible, machine-readable storage media, such as storage 422. The storage 422 may include any combinations of hard drives, read-only memory (ROM), random access memory (RAM), RAM drives, flash drives, optical drives, cache memory, and the like. The storage 422 may include a unified threat manager (UTM) 424, which performs the techniques described herein.

[0033] The storage 422 may also include a hash lookup table 426. The hash lookup table 426 includes an entry for each resource determined to contain malware. The UTM 424 uses the hash lookup table 426 to determine if resources requested by the clients 404 have been previously determined to contain malware. Further, the UTM 424 may add entries to the hash lookup table 426 whenever a requested resource is determined to contain malware.

[0034] Fig. 5 is a block diagram of an example system 500 for detecting malware. The system is generally referred to by the reference number 500. The system 500 is a computing device that includes a processor 502 and a memory 504. The memory 504 includes code 506 to determine a resource comprises malware. The memory 504 additionally includes code 508 to generate an entry in a hash table for the resource. Further, the memory 504 includes code 510 to determine, in response to a request to download a resource, whether the resource has previously been determined to comprise malware. Also, the memory 504 includes code 512 to determine, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination. Further, the memory 504 includes

code 514 to terminate the request to download the resource if the resource has not changed.

[0035] Fig. 6 is a block diagram showing an example non-transitory, tangible computer-readable medium 600 that stores code for detecting malware. The non-transitory, tangible computer-readable medium is generally referred to by the reference number 600.

[0036] The non-transitory, tangible computer-readable medium 600 may correspond to any typical storage device that stores computer-implemented instructions, such as programming code or the like. For example, the non-transitory, tangible computer-readable medium 600 may include one or more of a non-volatile memory, a volatile memory, and/or one or more storage devices.

[0037] Examples of non-volatile memory include, but are not limited to, electrically erasable programmable read only memory (EEPROM) and read only memory (ROM). Examples of volatile memory include, but are not limited to, static random access memory (SRAM), and dynamic random access memory (DRAM). Examples of storage devices include, but are not limited to, hard disks, compact disc drives, digital versatile disc drives, and flash memory devices.

[0038] A processor 602 generally retrieves and executes the computer-implemented instructions stored in the non-transitory, tangible computer-readable medium 600 for graph-based merger of detections. A unified threat manager 604 may detect malware for resources previously requested. In examples of the claimed subject matter. The unified threat manager 604 maintains a hash lookup table that contains an entry for all requested resources found to contain malware. Subsequently, whenever a resource is requested, the UTM 604 checks the hash lookup table to determine if the resource was previously determined to contain malware. If so, the current Etag of the resource is compared to the Etag at the time the resource was determined to contain malware. If the Etag is not changed, the resource is determined to still contain malware, and the connection with the server providing the resource is reset.

[0039] Although shown as contiguous blocks, the software components can be stored in any order or configuration. For example, if the computer-readable medium 600 is a hard drive, the software components can be stored in non-contiguous, or even overlapping, sectors.

[0040] Fig. 7 is a block diagram showing an example non-transitory, tangible computer-readable medium 700 that stores code for detecting malware. The non-transitory, tangible computer-readable medium is generally referred to by the reference number 700. The medium 700 includes code 702 to determine a resource comprises malware. The medium 700 also includes code 704 to generate an entry in a hash table for the resource, wherein generating the entry comprises determining an absolute uniform resource locator (URL) for the resource, and determining an Etag for the resource, wherein the entry comprises the absolute URL and the Etag. Additionally, the medium 700 includes code 706 to determine, in response to a request to download the resource, whether the resource has previously been determined to comprise malware. Further, the medium 700 includes code 708 to determine, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination. Also, the medium 700 includes code 710 to terminate the request to download the resource if the resource has not changed.

[0041] The present techniques are not restricted to the particular details listed herein. Indeed, those skilled in the art having the benefit of this disclosure will appreciate that many other variations from the foregoing description and drawings may be made within the scope of the present techniques. Accordingly, it is the following claims including any amendments thereto that define the scope of the present techniques.

CLAIMS

What is claimed is:

1. A method for detecting malware, comprising:
determining, in response to a request to download a resource, whether the resource has previously been determined to comprise malware;
determining, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination; and
terminating the request to download the resource if the resource has not changed.

2. The method of claim 1, comprising:
determining the resource comprises malware; and
generating an entry in a hash table for the resource.

3. The method of claim 2, wherein determining the resource comprises malware comprises one of:
signature scanning the resource;
performing a hash lookup of an entire file comprising the resource; and
monitoring the resource in a sandbox execution environment.

4. The method of claim 2, wherein generating the entry comprises:
determining an absolute uniform resource locator (URL) for the resource; and
determining an Etag for the resource, wherein the entry comprises the absolute URL and the Etag.

5. The method of claim 2, wherein determining whether the resource has previously been determined to comprise malware comprises performing a lookup in the hash table, wherein the resource has previously been determined to comprise malware if the lookup is successful.

6. The method of claim 4, wherein determining whether the resource has previously been determined to comprise malware comprises performing a lookup in the hash table using the absolute URL, wherein the resource has previously been determined to comprise malware if the lookup is successful.

7. The method of claim 5, wherein determining whether the resource has changed comprises:
determining the Etag based on the successful lookup;
determining a current Etag for the resource;
comparing the Etag to the current Etag, wherein the resource has not changed if the Etag is equal to the current Etag.

8. The method of claim 6, wherein determining whether the resource has changed comprises:
determining the Etag based on the successful lookup;
determining a current Etag for the resource;
comparing the Etag to the current Etag, wherein the resource has not changed if the Etag is equal to the current Etag.

9. A system for detecting malware, comprising:
a processor; and
a memory comprising code that causes the processor to:
determine a resource comprises malware;
generate an entry in a hash table for the resource;
determine, in response to a request to download the resource, whether the resource has previously been determined to comprise malware;
determine, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination; and
terminate the request to download the resource if the resource has not changed.

10. The system of claim 9, wherein determining the resource comprises malware comprises one of:
signature scanning the resource;
performing a hash lookup of an entire file comprising the resource; and
monitoring the resource in a sandbox execution environment.

11. The system of claim 9, wherein generating the entry comprises:
determining an absolute uniform resource locator (URL) for the resource; and
determining an Etag for the resource, wherein the entry comprises the absolute URL and the Etag.

12. The system of claim 9, wherein determining whether the resource has previously been determined to comprise malware comprises performing a lookup in the hash table, wherein the resource has previously been determined to comprise malware if the lookup is successful.

13. The system of claim 11, wherein determining whether the resource has previously been determined to comprise malware comprises performing a lookup in the hash table using the absolute URL, wherein the resource has previously been determined to comprise malware if the lookup is successful.

14. The system of claim 13, wherein determining whether the resource has changed comprises:
determining the Etag based on the successful lookup;
determining a current Etag for the resource;
comparing the Etag to the current Etag, wherein the resource has not changed if the Etag is equal to the current Etag.

15. A non-transitory, tangible computer-readable medium, comprising code to direct a processor to:

determine a resource comprises malware;

generate an entry in a hash table for the resource, wherein generating the entry comprises:

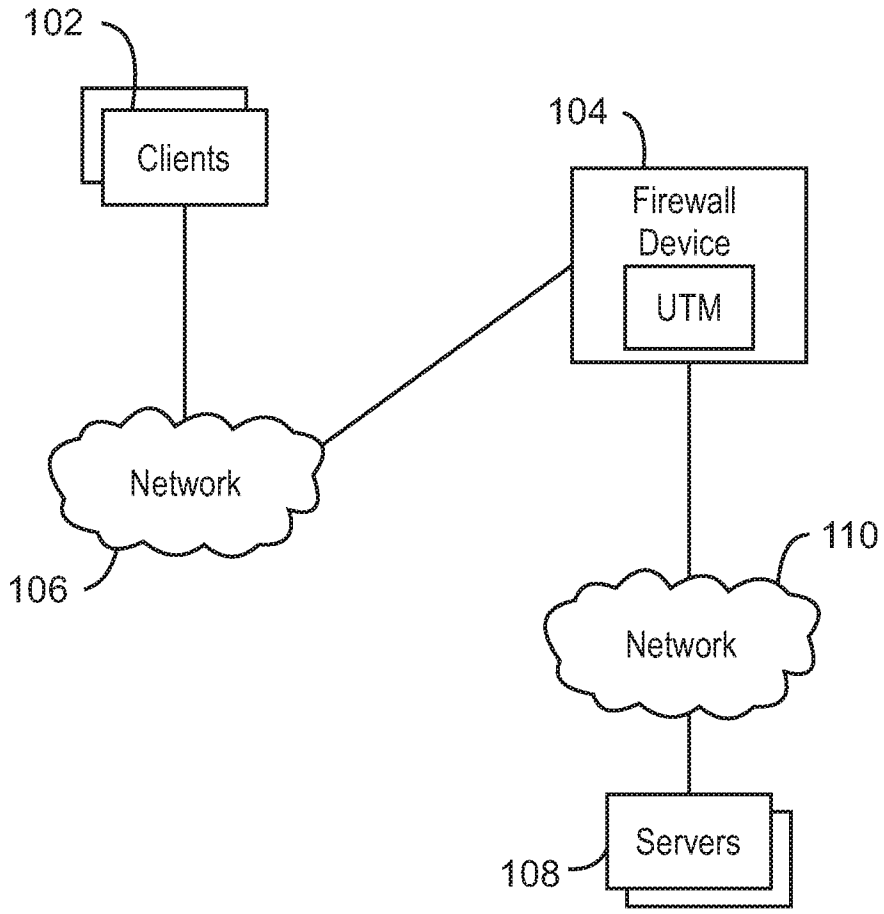
determining an absolute uniform resource locator (URL) for the resource; and

determining an Etag for the resource, wherein the entry comprises the absolute URL and the Etag;

determine, in response to a request to download the resource, whether the resource has previously been determined to comprise malware;

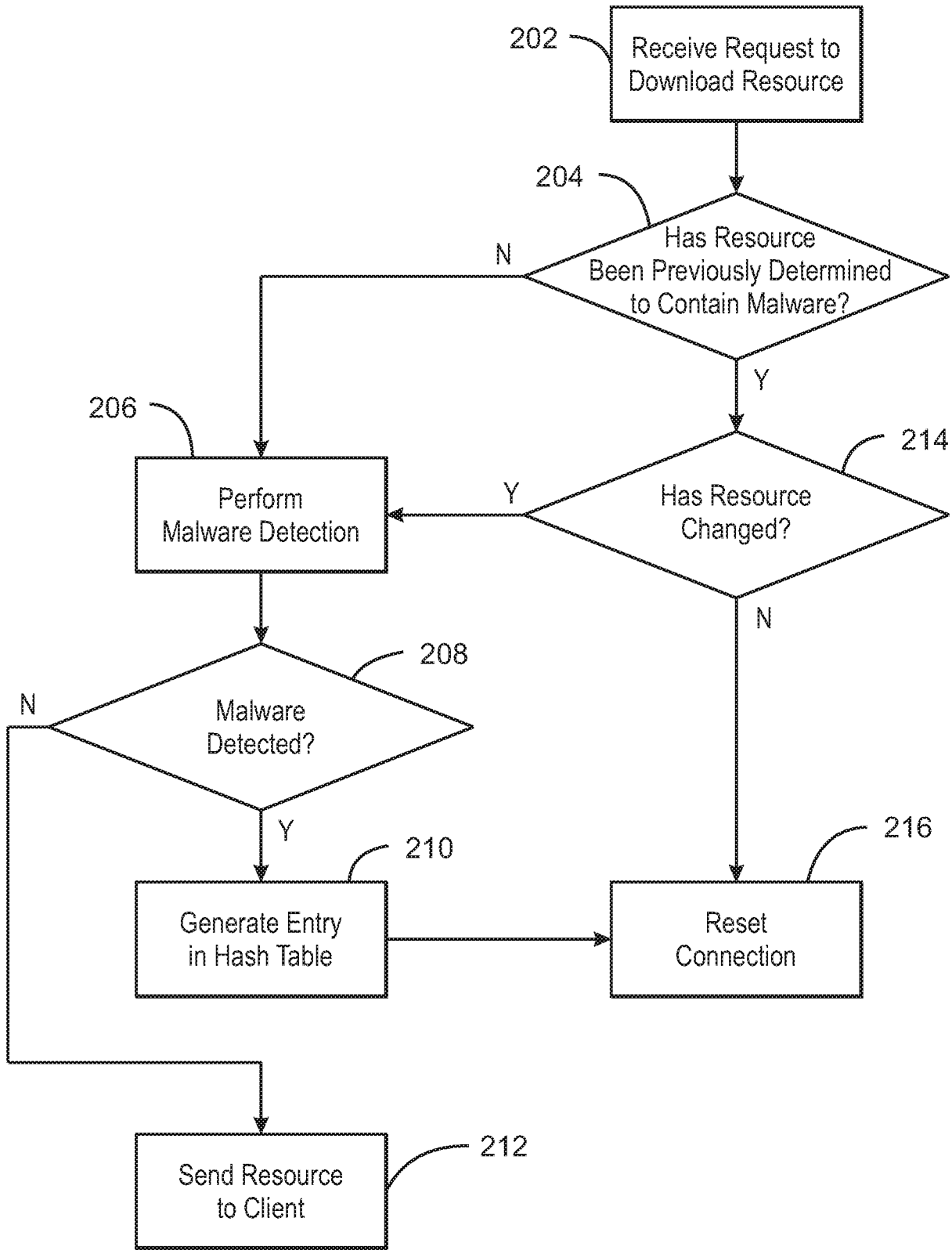
determine, if the resource has previously been determined to comprise malware, whether the resource has changed since the previous determination; and

terminate the request to download the resource if the resource has not changed.


102
Clients
104
Firewall
Device
UTM
Network
106
110
Network
Servers
108

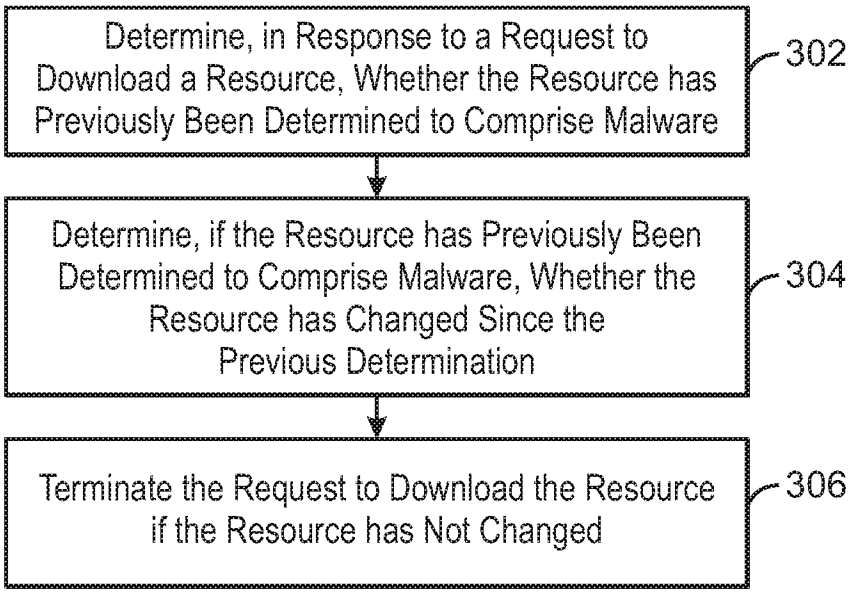
100

FIG. 1


202
Receive Request to Download Resource
204
Has Resource Been Previously Determined to Contain Malware?
N
Y
206
Perform Malware Detection
214
Has Resource Changed?
Y
N
208
Malware Detected?
N
Y
210
Generate Entry in Hash Table
216
Reset Connection
212
Send Resource to Client

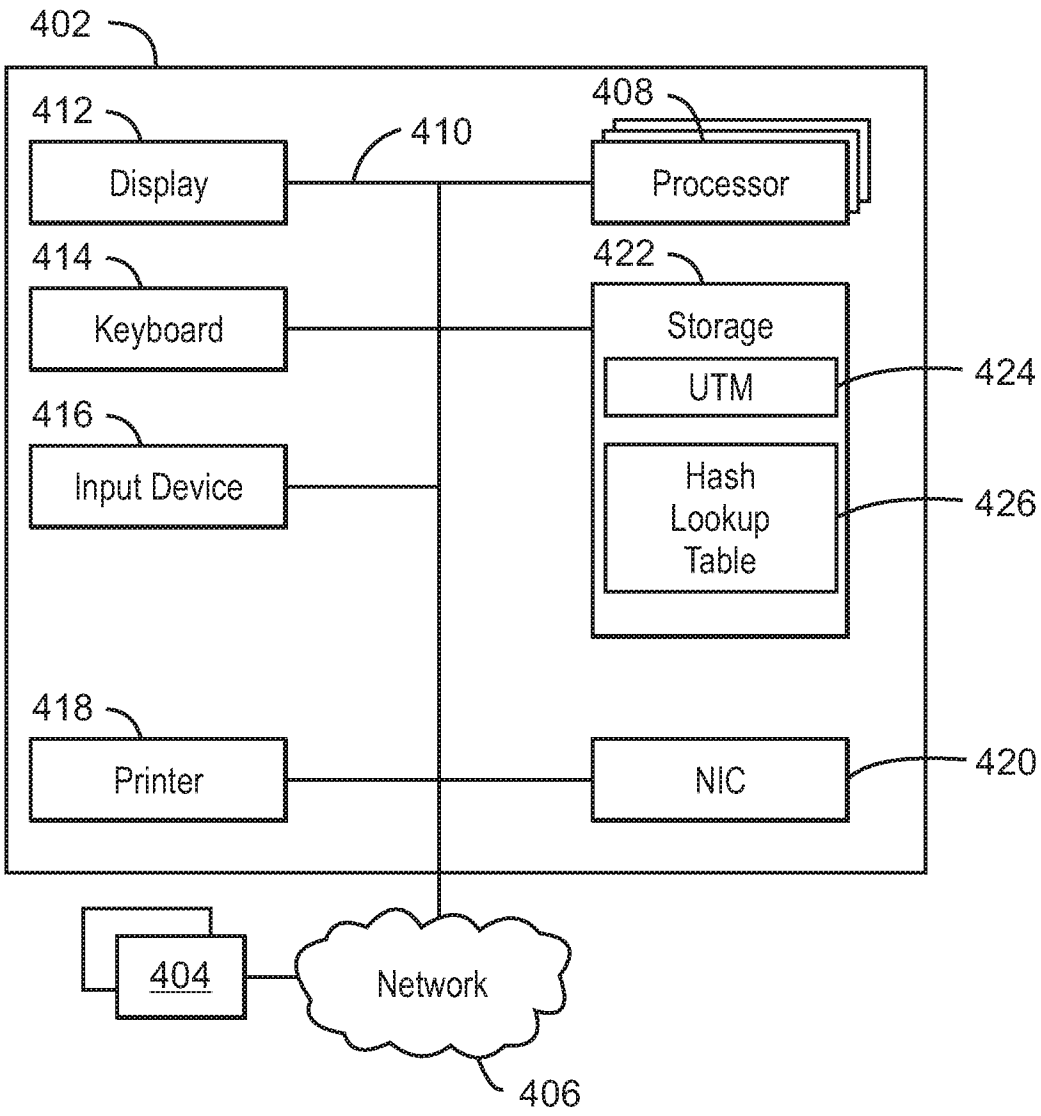
200

FIG. 2


Determine, in Response to a Request to Download a Resource, Whether the Resource has Previously Been Determined to Comprise Malware
302
Determine, if the Resource has Previously Been Determined to Comprise Malware, Whether the Resource has Changed Since the Previous Determination
304
Terminate the Request to Download the Resource if the Resource has Not Changed
306

300

FIG. 3


402
412
Display
410
408
Processor
414
Keyboard
422
Storage
UTM
424
416
Input Device
Hash
Lookup
Table
426
418
Printer
NIC
420
404
Network
406

400

FIG. 4

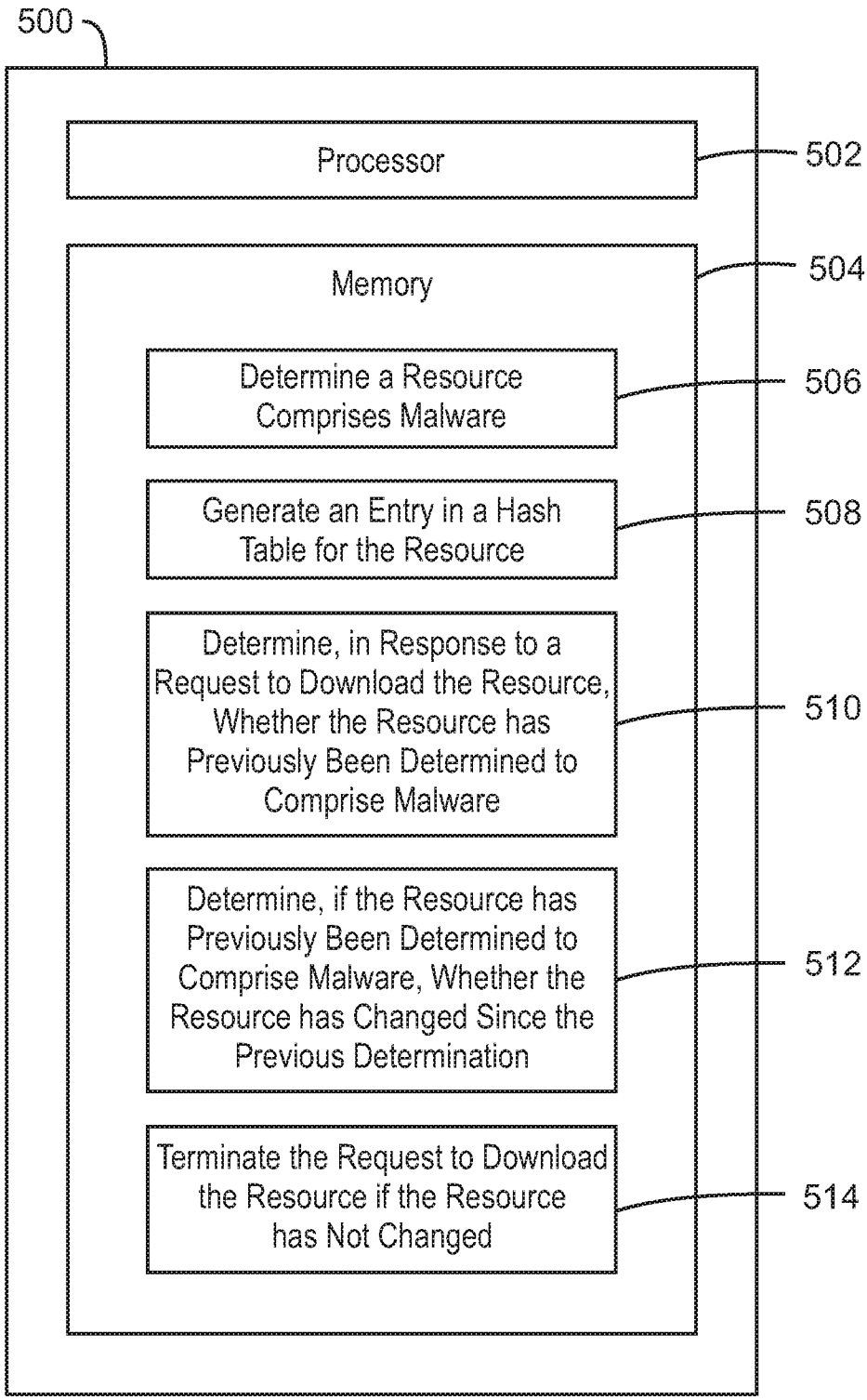

500
Processor
502
Memory
504
Determine a Resource Comprises Malware
506
Generate an Entry in a Hash Table for the Resource
508
Determine, in Response to a Request to Download the Resource, Whether the Resource has Previously Been Determined to Comprise Malware
510
Determine, if the Resource has Previously Been Determined to Comprise Malware, Whether the Resource has Changed Since the Previous Determination
512
Terminate the Request to Download the Resource if the Resource has Not Changed
514

FIG. 5

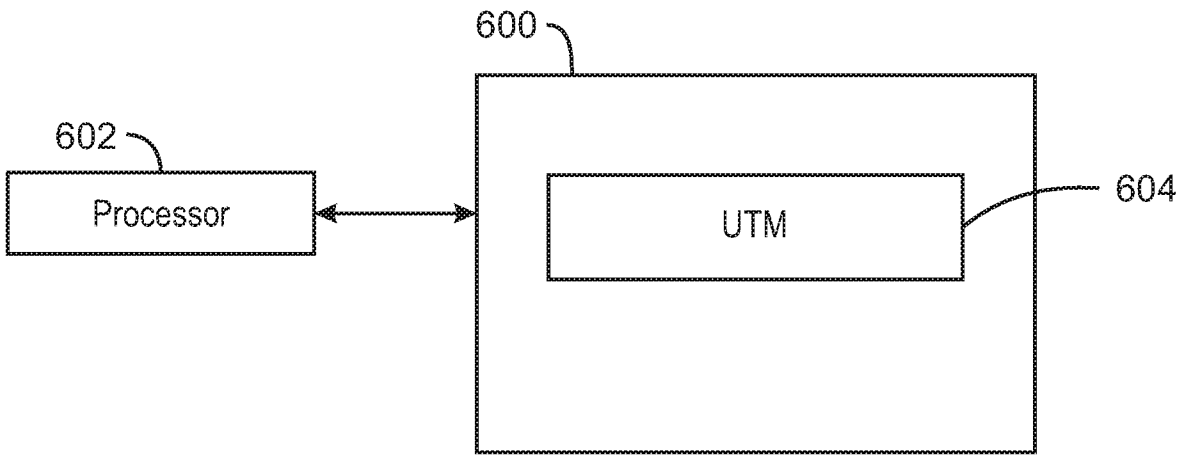


FIG. 6

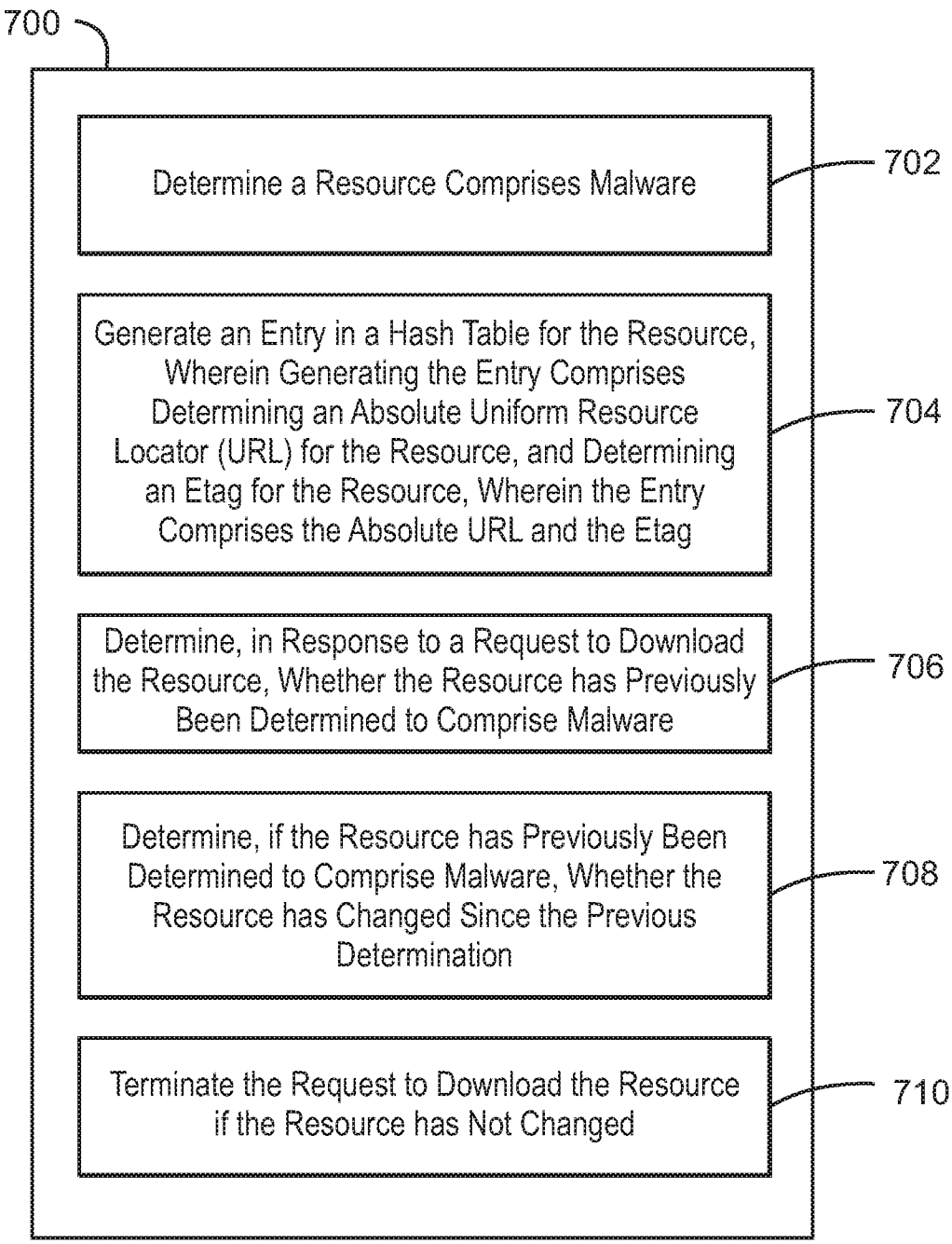

700
Determine a Resource Comprises Malware
702
Generate an Entry in a Hash Table for the Resource, Wherein Generating the Entry Comprises Determining an Absolute Uniform Resource Locator (URL) for the Resource, and Determining an Etag for the Resource, Wherein the Entry Comprises the Absolute URL and the Etag
704
Determine, in Response to a Request to Download the Resource, Whether the Resource has Previously Been Determined to Comprise Malware
706
Determine, if the Resource has Previously Been Determined to Comprise Malware, Whether the Resource has Changed Since the Previous Determination
708
Terminate the Request to Download the Resource if the Resource has Not Changed
710

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/014841

A. CLASSIFICATION OF SUBJECT MATTER

G06F 21/56(2013.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 21/56; G06F 11/30; G06F 11/00; G06F 21/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & keywords: download, resource, malware, Etag, detecting

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2011-0219450 A1 (MONTY D. MCDOUGAL et al.) 08 September 2011 See paragraphs [0017]-[0097]; and figures 1-6.	1-15
A	US 2013-0145470 A1 (MATTHEW RICHARD et al.) 06 June 2013 See paragraphs [0011]-[0047]; and figures 1-4.	1-15
A	US 2012-0185939 A1 (PAVEL TURBIN) 19 July 2012 See paragraphs [0039]-[0043]; and figures 1-2.	1-15
A	US 2011-0083186 A1 (JARNO NIEMELA et al.) 07 April 2011 See paragraphs [0044]-[0068]; and figures 1-2.	1-15
A	US 2011-0197281 A1 (CHRISTOPH ALME et al.) 11 August 2011 See paragraphs [0055]-[0116]; and figures 3-4.	1-15

☐ Further documents are listed in the continuation of Box C.

☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
12 September 2016 (12.09.2016)	**12 September 2016 (12.09.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
LEE, EUN KYU
Telephone No. +82-42-481-3580

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2016/014841

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2011-0219450 A1	08/09/2011	US 8863279 B2	14/10/2014
		WO 2011-112347 A2	15/09/2011
		WO 2011-112347 A3	03/11/2011
US 2013-0145470 A1	06/06/2013	AU 2012-347734 A1	13/06/2013
		AU 2012-347734 B2	09/10/2014
		CA 2856730 A1	13/06/2013
		CA 2856730 C	12/01/2016
		GB 2511691 A	10/09/2014
		GB 2511691 B	25/11/2015
		US 8510841 B2	13/08/2013
		WO 2013-086176 A1	13/06/2013
US 2012-0185939 A1	19/07/2012	EP 2663944 A1	20/11/2013
		EP 2663944 B1	26/08/2015
		US 8621634 B2	31/12/2013
		WO 2012-095348 A1	19/07/2012
US 2011-0083186 A1	07/04/2011	EP 2486507 A1	15/08/2012
		US 8590045 B2	19/11/2013
		WO 2011-042304 A1	14/04/2011
US 2011-0197281 A1	11/08/2011	US 8850584 B2	30/09/2014