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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
7 April 2016 (07.04.2016)

WIPO | PCT

(10) International Publication Number
WO 2016/053261 A1

(51) International Patent Classification:
H04L 12/26 (2006.01) *H04L 12/58* (2006.01)

(21) International Application Number:
PCT/US2014/058144

(22) International Filing Date:
29 September 2014 (29.09.2014)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).

(72) Inventors: **MUTHURAJAN, Sasi Siddharth**; 5555 Windward Pkwy, Alpharetta, Georgia 30004 (US). **SECHMAN, Ronald Joseph**; 5555 Windward Pkwy, Alpharetta, Georgia 30004 (US).

(74) Agents: **LEE, Rachel Jeong-Eun** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 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))*

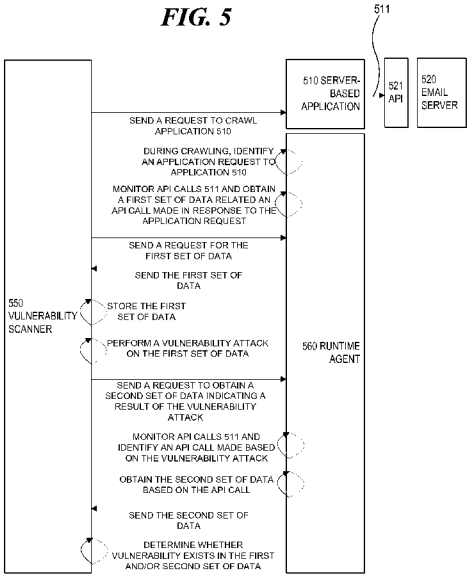
— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

Published:

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

(54) Title: DETECTION OF EMAIL-RELATED VULNERABILITIES

FIG. 5



(57) Abstract: Examples relate to detection of email-related vulnerabilities. The examples disclosed herein enable monitoring, at a runtime, application programming interface (API) calls made by a server-based application for the API calls related to at least one of a plurality of email protocols. A request to obtain a first set of data indicating a result of a vulnerability attack may be received from a vulnerability scanner. The examples disclosed herein enable identifying, at the runtime, an API call that has been made based on the vulnerability attack in response to receiving the request. The first set of data may be obtained, at the runtime, based on the API call. The examples disclosed herein further enable providing the first set of data to the vulnerability scanner to detect a vulnerability in the first set of data.

WO 2016/053261 A1

DETECTION OF EMAIL-RELATED VULNERABILITIES

BACKGROUND

[0001] Because the Internet enables rapid and widespread distribution of data, electronic correspondences such as electronic mail ("email") are becoming a popular means for communication. Electronic correspondences are widely used as a common mode of communication for both personal and professional use. Sensitive information such as medical information, financial information, legal documents, and even government correspondences are electronically exchanged over the Internet.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram depicting an example system comprising various components including a client computing device in communication with a server computing device for detecting email-related vulnerabilities.

[0004] FIG. 2 is a block diagram depicting an example server computing device for detecting email-related vulnerabilities.

[0005] FIG. 3 is a block diagram depicting an example machine-readable storage medium comprising instructions executable by a processor for detecting email-related vulnerabilities.

[0006] FIG. 4 is a block diagram depicting an example machine-readable storage medium comprising instructions executable by a processor for email-related vulnerabilities.

[0007] FIG. 5 is a flow diagram depicting an example data flow for detecting email-related vulnerabilities.

[0008] FIG. 6 is a flow diagram depicting an example method for detecting email-related vulnerabilities.

DETAILED DESCRIPTION

[0009] The following detailed description refers to the accompanying drawings. Wherever possible, the same reference numbers are used in the drawings and the following description to refer to the same or similar parts. It is to be expressly understood, however, that the drawings are for the purpose of illustration and description only. While several examples are described in this document, modifications, adaptations, and other implementations are possible. Accordingly, the following detailed description does not limit the disclosed examples. Instead, the proper scope of the disclosed examples may be defined by the appended claims.

[0010] Because the Internet enables rapid and widespread distribution of data, electronic correspondences such as electronic mail ("email") are becoming a popular means for communication. Electronic correspondences are widely used as a common mode of communication for both personal and professional use. Sensitive information including medical information, financial information, legal documents, and even government correspondences are electronically exchanged over the Internet. Electronic correspondences may be vulnerable to many types of attacks, which may seriously endanger the security of sensitive information. One example of a vulnerability related to email communications may be a vulnerability against an email header injection (e.g., similar to Carriage Return Line Feed (CRLF) injection). This type of vulnerability may occur in a header section of an email. The header section contains various fields such as "To," "From," "Subject," "CC," etc. If the "Subject" field is not validated, the email header injection may be used to inject another "To" field below the "Subject" field. In this way, the email at issue may be sent to the original

recipient (e.g., intended by the sender of the email) and the injected recipient (e.g., unintended by the sender of the email).

[0011] The vulnerability related to the email header injection may be detected through manual testing. A human tester may try to inject an additional "To" field that would send an email to the tester himself. Whether the tester actually receives that email is the only way to determine the presence or absence of the vulnerability against the email header injection. Even when an automated tool is used, the presence of the vulnerability may be confirmed only when the email is received by the injected recipient. In other words, the email header injection vulnerability may be detected only at the recipient's end. Moreover, these approaches may be applicable to the vulnerability related to the email header injection but not other types of email-related vulnerabilities. Other types of email-related vulnerabilities may relate to an email command injection, a sensitive information disclosure, an insecure email connection, and various other types of email-related vulnerabilities.

[0012] Examples disclosed herein address these technical issues by detecting email-related vulnerabilities that may be present in any data communicated using at least one email protocol, including, but not being limited to, outgoing messages, incoming messages, email commands or instructions to an email server, and/or email connections. The email protocols may be Transmission Control Protocol and the Internet Protocol ("TCP/IP protocols") used for mail delivery, which may include, but not be limited to, Simple Mail Transfer Protocol (SMTP), Internet Message Access Protocol (IMAP), and/or Post Office Protocol (POP3).

[0013] The examples disclosed herein enable monitoring, at a runtime, application programming interface (API) calls made by a server-based application for the API calls related to at least one of a plurality of email protocols. A request to obtain a first set of data indicating a result of a vulnerability attack may be received from a vulnerability scanner. The examples disclosed herein enable identifying, at the runtime, an API call that has been made based on the vulnerability attack in response to receiving the request. The first set of data may be obtained, at the runtime, based on the API call. The examples disclosed herein

further enable providing the first set of data to the vulnerability scanner to detect a vulnerability in the first set of data.

[0014] FIG. 1 is a block diagram depicting an example system 100 comprising various components including a client computing device in communication with a server computing device for detecting email-related vulnerabilities.

[0015] The various components may include client computing devices 140 (illustrated as 140A, 140B, ..., 140N) and a server computing device 130. Each client computing device 140A, 140B, ..., 140N may communicate requests to and/or receive responses from server computing device 130. Server computing device 130 may receive and/or respond to requests from client computing devices 140. Client computing devices 140 may be any type of computing device providing a user interface through which a user can interact with a software application. For example, client computing devices 140 may include a laptop computing device, a desktop computing device, an all-in-one computing device, a tablet computing device, a mobile phone, an electronic book reader, a network-enabled appliance such as a "Smart" television, and/or other electronic device suitable for displaying a user interface and processing user interactions with the displayed interface. While server computing device 130 is depicted as a single computing device, server computing device 130 may include any number of computing devices.

[0016] The various components (e.g., components 129, 130, and/or 140) depicted in FIG. 1 may be coupled to at least one other component via a network 50. Network 50 may comprise any infrastructure or combination of infrastructures that enable electronic communication between the components. For example, network 50 may include at least one of the Internet, an intranet, a PAN (Personal Area Network), a LAN (Local Area Network), a WAN (Wide Area Network), a SAN (Storage Area Network), a MAN (Metropolitan Area Network), a wireless network, a cellular communications network, a Public Switched Telephone Network, and/or other network. According to various implementations, system 100 and the various components described herein may be implemented in hardware and/or a combination

of hardware and programming that configures hardware. Furthermore, in FIG. 1 and other Figures described herein, different numbers of components or entities than depicted may be used.

[0017] Server computing device 130 may comprise a vulnerability test engine 121, a test result request engine 122, a test result receipt engine 123, a vulnerability detect engine 124, and/or other engines. The term "engine", as used herein, refers to a combination of hardware and programming that performs a designated function. As is illustrated respect to FIGS. 3-4, the hardware of each engine, for example, may include one or both of a processor and a machine-readable storage medium, while the programming is instructions or code stored on the machine-readable storage medium and executable by the processor to perform the designated function. All or part of the functionality of engines (illustrated in FIGS. 1-2) may co-exist in a single server computing device or be distributed among multiple server computing devices.

[0018] Vulnerability test engine 121 may perform a vulnerability test on a first set of data. The first set of data may include, for example, outgoing messages (e.g., an outgoing email, an outgoing Short Message Service (SMS), an outgoing Multimedia Messaging Service (MMS), and/or other outgoing electronic correspondences), incoming messages (e.g., an incoming email, an incoming Short Message Service (SMS), an incoming Multimedia Messaging Service (MMS), and/or other incoming electronic correspondences), application requests made to a server-based application (e.g., a request to send a message such as an email, SMS, etc., a request to retrieve a message such as an email, SMS, etc., a request to open a connection with an email server, etc.), instructions to an email server, and/or other information that may be retrieved from API calls made by the server-based application in response to various application requests.

[0019] In some implementations, the first set of data may be collected by a runtime agent that may monitor API calls made by the server-based application in response to a crawl request, which is discussed in detail with respect to FIG. 3. In some implementations, the first set of data may be a set of training data that may be

used to test various email-related vulnerabilities in the server-based application. The first set of data may be stored in a data storage (e.g., a data storage 129).

[0020] The server-based application may be, for example, a web application that allows a user to access and/or interact with their email through the use of a web browser. Examples of a web-based email application may include Gmail and Yahoo! Mail. The web application may be in communication with at least one email server such as a SMTP server (e.g., for sending emails), an IMAP server (e.g., for sending and/or retrieving emails), a POP3 server (e.g., for sending and/or retrieving emails), or other types of email servers.

[0021] The vulnerability test may include an active test and/or a passive test. For example, the active test may include various hacking attacks such as a header injection, a command injection, and/or other hacking attacks that may be made to the first set of data. A result of the hacking attacks (e.g., a second set of data) may be inspected to determine whether a vulnerability is detected in the result of the hacking attacks. On the other hand, the passive test may include, for example, a test to determine whether sensitive information (e.g., that is unencrypted) is found in the first set of data (e.g., outgoing and/or incoming messages). Another example of the passive test may be a test to determine whether a connection with an email server is insecure based on the first set of data (e.g., information related to an application request to open a connection with the email server).

[0022] The email header injection (e.g., similar to Carriage Return Line Feed (CRLF) injection) may manipulate a header section of an outgoing email. The header section contains various fields such as “To,” “From,” “Subject,” “CC,” etc. If the “Subject” field is not validated, the header injection may be used to inject another “To” field below the “Subject” field. In this way, the email at issue may be sent to the original recipient (e.g., intended by the sender of the email) and the injected recipient (e.g., unintended by the sender of the email). In the case of the email header injection, if the result of the email header injection attack, when inspected, indicates that the header has been indeed modified by the injection attack to include the

injected recipient, it may be determined that a vulnerability against the email header injection is detected. Such a vulnerability can be used as a spam relay that makes the email to be sent from the victim's server, making it difficult to trace back to the spammer.

[0023] The email command injection (e.g., similar to a CRLF injection) may inject a command (e.g., SMTP, IMAP, and/or POP3 commands made to corresponding email servers) into an application request to execute the command in the corresponding email server. In the case of the email command injection, if the command is executed in the email server, a vulnerability against the email command injection may be detected.

[0024] The first set of data may contain sensitive information in cleartext (e.g., unprotected or unencrypted). Such a sensitive information disclosure may occur, for example, when a password (e.g., a reset password) is included in the first set of data (e.g., outgoing and/or incoming messages). The first set of data may be inspected to determine whether such sensitive information is found. If such sensitive information is found, a vulnerability may be detected.

[0025] A vulnerability test to determine whether a connection with an email server is insecure may be performed on the first set of data. The first set of data may include an application request to open a connection with the email server and/or other information related to an application request to open a connection with the email server such as information from the API call (e.g., connection opening calls that may indicate the email server's port and/or protocol being used to connect) made by the server-based application in response to the application request. In order to provide transport layer security for connections to email servers, their corresponding Secure Sockets Layer (SSL) versions (e.g., SMTPS, IMAPS, POP3S, etc.) have been introduced. One way of testing the security of the connection may, therefore, be to determine whether the connection is made over SSL (e.g., secure connection) or non-SSL (e.g., insecure connection).

[0026] Test result request engine 122 may generate a request to obtain a second set of data indicating a result of the vulnerability test (e.g., active tests) and/or sends the request to obtain the second set of data to a runtime agent. The runtime agent may monitor, at a runtime, API calls made by the server-based application for the API calls related to at least one of a plurality of email protocols. For example, the runtime agent may identify an API call related to (and/or made by the server-based application in response to) the vulnerability test (performed by vulnerability test engine 121). The runtime agent may retrieve the second set of data based on this identified API call. The second set of data may include, for example, incoming and/or outgoing messages (e.g., raw emails, SMS messages, etc.) made in response to the header injection attack, a result indicating whether the injected command has been executed by the email server in response to the command injection attack, and/or other information that may be retrieved from the API call.

[0027] Test result receipt engine 123 may receive, from the runtime agent, the second set of data. The second set of data, in some implementations, may be stored in the data storage (e.g., data storage 129).

[0028] Vulnerability detect engine 124 may determine whether a vulnerability is detected in the second set of data. In doing so, vulnerability detect engine 124 may inspect the second set of data to determine whether the vulnerability exists in the second set of data. For example, in the case of the email header injection, if the second set of data, when inspected, indicates that the header section has been indeed modified by the header injection attack to include the injected recipient (e.g., that the additional recipient appears in the header section), it may be determined that a vulnerability against the email header injection is detected. In the case of the email command injection, if the second set of data indicates that the injected command has been executed in the email server, a vulnerability against the email command injection may be detected.

[0029] Vulnerability detect engine 124 may determine whether a vulnerability is detected in the first set of data. In doing so, vulnerability detect engine 124 may

inspect the first set of data to determine whether the vulnerability exists in the first set of data. For example, in the case of the sensitive information disclosure, vulnerability detect engine 124 may inspect the first set of data (e.g., a body section of the first set of data that comprises an email message) to determine whether sensitive information that is unencrypted is found in the first set of data. If such sensitive information is found, a vulnerability against the sensitive information disclosure may be detected. In another example, in the case of the insecure connection, vulnerability detect engine 124 may inspect the first set of data to determine whether a connection with an email server is insecure. The first set of data may include an application request to open a connection with the email server and/or other information related to an application request to open a connection with the email server such as information from the API call (e.g., connection opening calls that may indicate the email server's port and/or protocol being used to connect) made by the server-based application in response to the application request. In order to provide transport layer security for connections to email servers, their corresponding Secure Sockets Layer (SSL) versions (e.g., SMTPS, IMAPS, POP3S, etc.) have been introduced. One way of testing the security of the connection may, therefore, be to determine whether the connection is made over SSL (e.g., secure connection) or non-SSL (e.g., insecure connection).

[0030] In performing their respective functions, engines 121-124 may access data storage 129 and/or other suitable database(s). Data storage 129 may represent any memory accessible to server computing device 130 that can be used to store and retrieve data. Data storage 129 and/or other database may comprise random access memory (RAM), read-only memory (ROM), electrically-erasable programmable read-only memory (EEPROM), cache memory, floppy disks, hard disks, optical disks, tapes, solid state drives, flash drives, portable compact disks, and/or other storage media for storing computer-executable instructions and/or data. Server computing device 130 may access data storage 129 locally or remotely via network 50 or other networks.

[0031] Data storage 129 may include a database to organize and store data. Database 129 may be, include, or interface to, for example, an Oracle™ relational database sold commercially by Oracle Corporation. Other databases, such as Informix™, DB2 (Database 2) or other data storage, including file-based (e.g., comma or tab separated files), or query formats, platforms, or resources such as OLAP (On Line Analytical Processing), SQL (Structured Query Language), a SAN (storage area network), Microsoft Access™, MySQL, PostgreSQL, HSpace, Apache Cassandra, MongoDB, Apache CouchDB™, or others may also be used, incorporated, or accessed. The database may reside in a single or multiple physical device(s) and in a single or multiple physical location(s). The database may store a plurality of types of data and/or files and associated data or file description, administrative information, or any other data.

[0032] FIG. 2 is a block diagram depicting an example server computing device 230 for detecting email-related vulnerabilities. Server computing device 230, although depicted as a single device, may include any number of computing devices. In the example depicted in FIG. 2, a vulnerability scanner 250 and a runtime agent 260 may co-exist in a single server computing device or may exist in different server computing devices.

[0033] Vulnerability scanner 250 may comprise a vulnerability test engine 251, a test result request engine 252, a test result receipt engine 253, and a vulnerability detect engine 254, and/or other engines. Engines 251-254 represent engines 121-124, respectively. In some implementations, vulnerability scanner 250 may be, for example a blackbox vulnerability scanner that may be used to find security vulnerabilities in web applications in an automated fashion. The blackbox vulnerability scanner may test any web application regardless of the server-side language. In other words, the blackbox vulnerability scanner may automatically test the server-side application for security vulnerabilities, without access to source code used to build the application.

[0034] Runtime agent 260 may comprise an API calls monitor engine 261, a test result request receipt engine 262, a test result provide engine 263, and/or other engines. Runtime agent 260 may be responsible for observing, at a runtime, the values that are being passed to and from various functions in the server-based application including application requests to the application and/or API calls made by the application.

[0035] API calls monitor engine 261 may monitor API calls made by the server-based application for the API calls related to at least one of the plurality of email protocols. During monitoring, API calls monitor engine 261 may identify the API calls that are being sent using at least one of the plurality of email protocols. For example, API calls monitor engine 261 may identify an API call made based on at least one of SMTP, IMAP, POP3, and other email-specific protocols.

[0036] Test result request receipt engine 262 may receive, from vulnerability scanner 250 (e.g., test result request engine 252), a request to obtain a set of data indicating a result of a vulnerability test (e.g., an active test such as a header injection and a command injection). In response to the request, test result request receipt engine 262 may identify an API call related to (and/or made by the server-based application in response to) the vulnerability test (performed by vulnerability test engine 251). The runtime agent may retrieve the set of data indicating the result of the vulnerability test based on this identified API call. The set of data may include, for example, incoming and/or outgoing messages (e.g., raw emails, SMS messages, etc.) made in response to the header injection attack, a result indicating whether the injected command has been executed by the email server in response to the command injection attack, and/or other information that may be retrieved from the API call.

[0037] Test result provide engine 263 may provide the set of data indicating the result of the vulnerability test to vulnerability scanner 250 to detect a vulnerability in the set of data (by vulnerability detect engine 254).

[0038] FIG. 3 is a block diagram depicting an example machine-readable storage medium 310 comprising instructions executable by a processor for detecting email-related vulnerabilities.

[0039] In the foregoing discussion, engines 261-263 were described as combinations of hardware and programming. Engines 261-263 may be implemented in a number of fashions. Referring to FIG. 3, the programming may be processor executable instructions 321-323 stored on a machine-readable storage medium 310 and the hardware may include a processor 311 for executing those instructions. Thus, machine-readable storage medium 310 can be said to store program instructions or code that when executed by processor 311 implements functionality of engines 261-263.

[0040] In FIG. 3, the executable program instructions in machine-readable storage medium 310 are depicted as API calls monitoring instructions 321, attack result request obtaining instructions 322, and attack result providing instructions 323. Instructions 321-323 represent program instructions that, when executed, cause processor 311 to implement functionality of engines 261-263, respectively.

[0041] FIG. 4 is a block diagram depicting an example machine-readable storage medium 410 comprising instructions executable by a processor for detecting email-related vulnerabilities.

[0042] In the foregoing discussion, engines 121-124 were described as combinations of hardware and programming. Engines 121-124 may be implemented in a number of fashions. Referring to FIG. 4, the programming may be processor executable instructions 421-424 stored on a machine-readable storage medium 410 and the hardware may include a processor 411 for executing those instructions. Thus, machine-readable storage medium 410 can be said to store program instructions or code that when executed by processor 411 implements functionality of engines 121-124.

[0043] In FIG. 4, the executable program instructions in machine-readable storage medium 410 are depicted as attack performing instructions 421, attack result

requesting instructions 422, attach result obtaining instructions 423, and vulnerability determining instructions 424. Instructions 421-424 represent program instructions that, when executed, cause processor 411 to implement functionality of engines 121-124, respectively.

[0044] Machine-readable storage medium 310 (or machine-readable storage medium 410) may be any electronic, magnetic, optical, or other physical storage device that contains or stores executable instructions. In some implementations, machine-readable storage medium 310 (or machine-readable storage medium 410) may be a non-transitory storage medium, where the term "non-transitory" does not encompass transitory propagating signals. Machine-readable storage medium 310 (or machine-readable storage medium 410) may be implemented in a single device or distributed across devices. Likewise, processor 311 (or processor 411) may represent any number of processors capable of executing instructions stored by machine-readable storage medium 310 (or machine-readable storage medium 410). Processor 311 (or processor 411) may be integrated in a single device or distributed across devices. Further, machine-readable storage medium 310 (or machine-readable storage medium 410) may be fully or partially integrated in the same device as processor 311 (or processor 411), or it may be separate but accessible to that device and processor 311 (or processor 411).

[0045] In one example, the program instructions may be part of an installation package that when installed can be executed by processor 311 (or processor 411) to implement functionality of engines 261-263 (or engines 121-124). In this case, machine-readable storage medium 310 (or machine-readable storage medium 410) may be a portable medium such as a floppy disk, CD, DVD, or flash drive or a memory maintained by a server from which the installation package can be downloaded and installed. In another example, the program instructions may be part of an application or applications already installed. Here, machine-readable storage medium 310 (or machine-readable storage medium 410) may include a hard disk, optical disk, tapes, solid state drives, RAM, ROM, EEPROM, or the like.

[0046] Processor 311 may be at least one central processing unit (CPU), microprocessor, and/or other hardware device suitable for retrieval and execution of instructions stored in machine-readable storage medium 310. Processor 311 may fetch, decode, and execute program instructions 321-323, and/or other instructions. As an alternative or in addition to retrieving and executing instructions, processor 311 may include at least one electronic circuit comprising a number of electronic components for performing the functionality of at least one of instructions 321-323, and/or other instructions.

[0047] Processor 411 may be at least one central processing unit (CPU), microprocessor, and/or other hardware device suitable for retrieval and execution of instructions stored in machine-readable storage medium 410. Processor 411 may fetch, decode, and execute program instructions 421-424, and/or other instructions. As an alternative or in addition to retrieving and executing instructions, processor 411 may include at least one electronic circuit comprising a number of electronic components for performing the functionality of at least one of instructions 421-424, and/or other instructions.

[0048] FIG. 5 is a flow diagram depicting an example data flow 500 for detecting email-related vulnerabilities.

[0049] A vulnerability scanner 550, a server-based application 510, a runtime agent 560, and an email server 520 (and its API 521) may co-exist in a single server computing device or may exist in different server computing devices. For example, server-based application 510, runtime agent 560, and email server 520 may be located in a first server computing device while vulnerability scanner 550 may exist in a second server computing device in communication with the first server computing device.

[0050] Vulnerability scanner 550 may send a request to crawl server–based application 510, which may cause crawling of server-based application 510 (e.g., following links, executing scripts, etc.). During crawling, runtime agent 560 may identify an application request to server-based application 510. Examples of the

application request may include a request to send a message such as an email, SMS, etc., a request to retrieve a message such as an email, SMS, etc., and/or a request to open a connection with an email server.

[0051] Runtime agent 560 may monitor, at a runtime, API calls 511 (e.g., made by server-based application 510 to API 521 of email server 520) to identify API calls 511 related to at least one of a plurality of email-protocols. Runtime agent 560 may obtain a first set of data related to an API call made in response to the identified application request. For example, if the identified application request is an HTTP request to send an email, runtime agent 560 may monitor and/or identify an API call corresponding to the HTTP request to send the email. Runtime agent 560 may retrieve and/or obtain the first set of data (e.g., outgoing email, instructions to a SMTP server, etc.) based on the API call corresponding to the HTTP request to send the email. In another example, if the identified application request is a HTTP request to retrieve an email, runtime agent 560 may monitor and/or identify an API call corresponding to the HTTP request to retrieve the email. Runtime agent 560 may retrieve and/or obtain the first set of data (e.g., incoming email, instructions to an IMAP server, etc.) based on the API call corresponding to the HTTP request to retrieve the email. In yet another example, if the identified application request is a request to open a connection with an email server, runtime agent 560 may monitor and/or identify an API call corresponding to the request to open the connection with the email server. Runtime agent 560 may retrieve and/or obtain the first set of data (e.g., details of the SMTP connection) based on the API call corresponding to the request to open the connection with the email server.

[0052] Vulnerability scanner 550 may send a request for the first set of data to runtime agent 560. Runtime agent 560 may send the first set of data to vulnerability scanner 550 that may store the first set of data in a data storage coupled to vulnerability scanner 550 (e.g., data storage 129 of FIG. 1).

[0053] Vulnerability scanner 550 may perform a vulnerability attack on the first set of data. The vulnerability attack may comprise various hacking attacks such as a

header injection, a command injection, and/or other hacking attacks that may be made to the first set of data.

[0054] Vulnerability scanner 550 may generate and/or send a request to obtain a second set of data indicating a result of the vulnerability attack to runtime agent 560. Runtime agent 560 may monitor, at a runtime, API calls made by server-based application 510 for the API calls related to at least one of the plurality of email protocols. For example, runtime agent 560 may identify an API call related to (and/or made by server-based application 510 in response to) the vulnerability attack (performed by vulnerability scanner 550). Runtime agent 560 may obtain the second set of data based on this identified API call. The second set of data may include, for example, incoming and/or outgoing messages (e.g., raw emails, SMS messages, etc.) made in response to the header injection attack, a result indicating whether the injected command has been executed by the email server in response to the command injection attack, and/or other information that may be obtained from the API call. Runtime agent 560 may send the second set of data to vulnerability scanner 550.

[0055] Vulnerability scanner 550 may determine whether a vulnerability exists in the first and/or second set of data.

[0056] In some implementations, vulnerability scanner 550 may determine whether a vulnerability exists in the first set of data. In doing so, vulnerability scanner 550 may inspect the first set of data to determine whether the vulnerability exists in the first set of data. For example, vulnerability scanner 550 may inspect the first set of data (e.g., an outgoing or incoming message) to determine whether sensitive information that is unencrypted is found in the first set of data. If such sensitive information is found, a vulnerability against the sensitive information disclosure may be detected. In another example, vulnerability scanner 550 may inspect the first set of data to determine whether a connection with an email server is insecure. The first set of data may include an application request to open a connection with the email server and/or other information related to an application request to open a connection

with the email server such as information from the API call (e.g., connection opening calls that may indicate the email server's port and/or protocol being used to connect) made by the server-based application in response to the application request. In order to provide transport layer security for connections to email servers, their corresponding Secure Sockets Layer (SSL) versions (e.g., SMTPS, IMAPS, POP3S, etc.) have been introduced. One way of testing the security of the connection may, therefore, be to determine whether the connection is made over SSL (e.g., secure connection) or non-SSL (e.g., insecure connection).

[0057] In some implementations, vulnerability scanner 550 may determine whether a vulnerability exists in the second set of data. In doing so, vulnerability scanner 550 may inspect the second set of data to determine whether the vulnerability exists in the second set of data. For example, in the case of the email header injection, if the second set of data, when inspected, indicates that the header section has been indeed modified by the header injection attack to include the injected recipient (e.g., that the additional recipient appears in the header section), it may be determined that a vulnerability against the email header injection is detected. In the case of the email command injection, if the second set of data indicates that the injected command has been executed in the email server, a vulnerability against the email command injection may be detected.

[0058] FIG. 6 is a flow diagram depicting an example method 600 for detecting email-related vulnerabilities. The various processing blocks and/or data flows depicted in FIG. 6 are described in greater detail herein. The described processing blocks may be accomplished using some or all of the system components described in detail above and, in some implementations, various processing blocks may be performed in different sequences and various processing blocks may be omitted. Additional processing blocks may be performed along with some or all of the processing blocks shown in the depicted flow diagrams. Some processing blocks may be performed simultaneously. Accordingly, method 600 as illustrated (and described in greater detail below) is meant be an example and, as such, should not

be viewed as limiting. Method 600 may be implemented in the form of executable instructions stored on a machine-readable storage medium, such as storage medium 310, and/or in the form of electronic circuitry.

[0059] Method 600 may start in block 621 where by a vulnerability scanner performs a vulnerability test on a first set of data. The first set of data may be collected by a runtime agent during an execution of a request to an application. In some implementations, the first set of data may be collected by the runtime agent in response to a crawl request from the vulnerability scanner (discussed herein in detail with respect to FIG. 3). The vulnerability test may include an active test and/or a passive test. For example, the active test may include various hacking attacks such as a header injection, a command injection, and/or other hacking attacks that may be made to the first set of data. A result of the hacking attacks (e.g., a second set of data) may be inspected to determine whether a vulnerability is detected in the result of the hacking attacks. On the other hand, the passive test may include, for example, a test to determine whether sensitive information (e.g., that is unencrypted) is found in the first set of data (e.g., outgoing and/or incoming messages). Another example of the passive test may be a test to determine whether a connection with an email server is insecure based on the first set of data (e.g., information related to an application request to open a connection with the email server).

[0060] In block 622, method 600 may include monitoring, by the runtime agent, API calls made by the application for the API calls related to at least one of a plurality of email protocols. In block 623, method 600 may include identifying, by the runtime agent, an API call generated in response to the vulnerability test. The runtime agent may retrieve a second set of data from this identified API call (block 624). The second set of data may include, for example, incoming and/or outgoing messages (e.g., raw emails, SMS messages, etc.) made in response to the header injection attack, a result indicating whether the injected command has been executed by the email server in response to the command injection attack, and/or other information that may be retrieved from the API call.

[0061] In block 625, method 600 may include determining whether a vulnerability is detected in the second set of data. In doing so, the vulnerability scanner may inspect the second set of data to determine whether the vulnerability exists in the second set of data. For example, in the case of the email header injection, if the second set of data, when inspected, indicates that the header section has been indeed modified by the header injection attack to include the injected recipient (e.g., that the additional recipient appears in the header section), it may be determined that a vulnerability against the email header injection is detected. In the case of the email command injection, if the second set of data indicates that the injected command has been executed in the email server, a vulnerability against the email command injection may be detected.

[0062] Referring back to FIG. 2, vulnerability test engine 251 may be responsible for implementing block 621. API calls monitor engine 261 may be responsible for blocks 622-623. Test result request receipt engine 262 may be responsible for implementing block 624. Vulnerability detect engine 254 may be responsible for implementing block 625.

[0063] The foregoing disclosure describes a number of example implementations for detection of email-related vulnerabilities. The disclosed examples may include systems, devices, computer-readable storage media, and methods for detection of email-related vulnerabilities. For purposes of explanation, certain examples are described with reference to the components illustrated in FIGS. 1-4. The functionality of the illustrated components may overlap, however, and may be present in a fewer or greater number of elements and components.

[0064] Further, all or part of the functionality of illustrated elements may co-exist or be distributed among several geographically dispersed locations. Moreover, the disclosed examples may be implemented in various environments and are not limited to the illustrated examples. Further, the sequence of operations described in connection with FIGS. 5-6 are examples and are not intended to be limiting. Additional or fewer operations or combinations of operations may be used or may

vary without departing from the scope of the disclosed examples. Furthermore, implementations consistent with the disclosed examples need not perform the sequence of operations in any particular order. Thus, the present disclosure merely sets forth possible examples of implementations, and many variations and modifications may be made to the described examples. All such modifications and variations are intended to be included within the scope of this disclosure and protected by the following claims.

CLAIMS

1. A method for execution by a server computing device for detecting email-related vulnerabilities, the method comprising:

performing, by a vulnerability scanner, a vulnerability test on a first set of data, wherein the first set of data is collected by a runtime agent during an execution of a request to an application running on the server computing device;

monitoring, by the runtime agent, application programming interface (API) calls made by the application for the API calls related to at least one of a plurality of email protocols;

identifying, by the runtime agent, an API call generated in response to the vulnerability test;

retrieving, by the runtime agent, a second set of data from the API call; and

determining, by the vulnerability scanner, whether a vulnerability is detected in the second set of data.

2. The method of claim 1, wherein the request to the application comprises at least one of a request to send an email, a request to read an email, and a request to open a connection with an email server.

3. The method of claim 1, wherein performing the vulnerability test on the first set of data comprises:

performing a hacking attack on the first set of data, wherein the hacking attack comprises at least one of:

a header injection attack that manipulates a header section of a raw email, wherein the first set of data comprises the raw email; and

a command injection attack that injects a command into the request to the application to execute the command in an email server.

4. The method of claim 1, further comprising:

determining, by the vulnerability scanner, whether the vulnerability is detected in the first set of data.

5. The method of claim 4, wherein determining, by the vulnerability scanner, whether the vulnerability is detected in the first set of data comprises at least one of:

inspecting, by the vulnerability scanner, a body section of the first set of data that comprises a raw email to determine whether the vulnerability is present in the body section of the raw email; and

inspecting, by the vulnerability scanner, the first set of data comprising information related to a request to open a connection with an email server to determine whether the connection is insecure.

6. A non-transitory machine-readable storage medium comprising instructions executable by a processor of a server computing device for detecting email-related vulnerabilities, the machine-readable storage medium comprising:

instructions to monitor, at a runtime, application programming interface (API) calls made by a server-based application for the API calls related to at least one of a plurality of email protocols;

instructions to receive, from a vulnerability scanner, a request to obtain a first set of data indicating a result of a vulnerability attack;

in response to receiving the request, instructions to identify, at the runtime, an API call that has been made based on the vulnerability attack;

instructions to obtain, at the runtime, the first set of data based on the API call; and

instructions to provide the first set of data to the vulnerability scanner to detect a vulnerability in the first set of data.

7. The non-transitory machine-readable storage medium of claim 6, further comprising:

instructions to obtain, at the runtime, a second set of data related to an API call made in response to an application request to the server-based application, wherein the vulnerability scanner performs the vulnerability attack on the second set of data.

8. The non-transitory machine-readable storage medium of claim 7, wherein the vulnerability attack comprises at least one of:

a first injection attack that, when the first injection attack is successful, modifies a header section of the first set of data, wherein the first set of data includes an outgoing message; and

a second injection attack that, when the second injection attack is successful, causes an email server to execute a set of commands that have been injected into the application request.

9. The non-transitory machine-readable storage medium of claim 8, wherein the vulnerability scanner inspects the second set of data to determine whether the vulnerability exists in the second set of data by at least one of:

inspecting a body section of the second set of data that comprises an outgoing message or an incoming message to determine whether sensitive information that is unprotected is included in the body section of the message; and

inspecting the second set of data that comprises information related to the application request to open a connection with an email server to determine whether the connection is insecure.

10. The non-transitory machine-readable storage medium of claim 7, wherein the application request comprises at least one of a request to send a message, a request to read a message, and a request to open a connection with an email server.

11. A system for detecting email-related vulnerabilities comprising:

a processor that:

performs a vulnerability test on a first set of data;

generates a request to obtain a second set of data indicating a result of the vulnerability test;

sends the request to a runtime agent that monitors, at a runtime, application programming interface (API) calls made by a server-based application for the API calls related to at least one of a plurality of email protocols, wherein the runtime agent retrieves the second set of data based on an API call related to the vulnerability test;

receives, from the runtime agent, the second set of data; and

determines whether a vulnerability is detected in the second set of data.

12. The system of claim 11, wherein performing the vulnerability test on the first set of data comprises:

performing an injection attack that injects an additional recipient into a header section of an outgoing email, wherein the first set of data comprises the outgoing email; and

inspecting the header section of the second set of data to determine whether the additional recipient appears in the header section of the second set of data.

13. The system of claim 11, wherein performing the vulnerability test on the first set of data comprises:

performing an injection attack that injects a command into an application request made to the server-based application, wherein the first set of data comprises the application request; and

determining whether the injected command has been executed in an email server.

14. The system of claim 11, the processor that:

inspects a body of the first set of data that comprises an email message to determine whether sensitive information that is unencrypted is found in the body of the email message.

15. The system of 11, wherein the vulnerability test is performed by a vulnerability scanner that automatically tests the server-based application for security vulnerabilities, without access to source code used to build the server-based application.

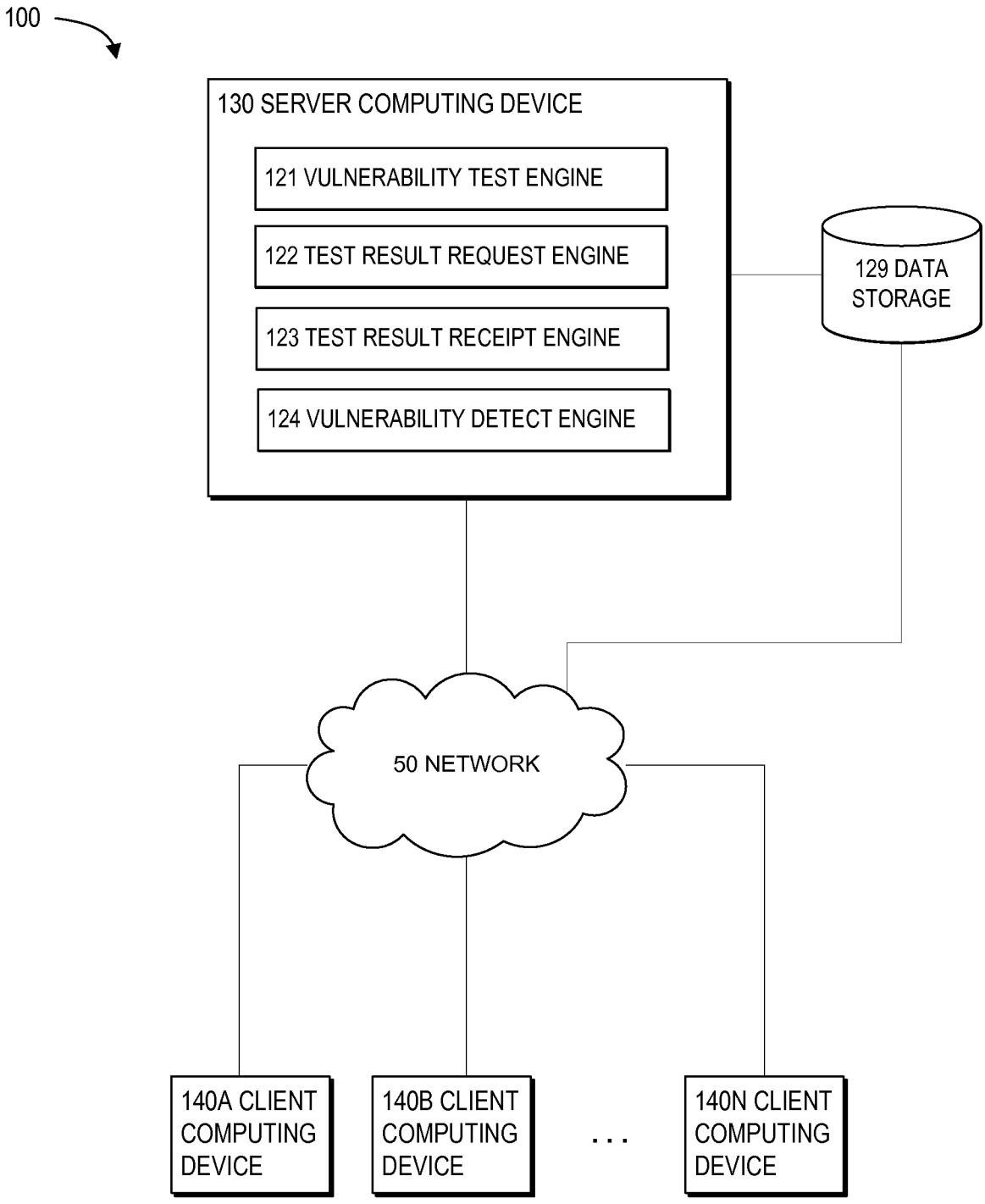

100
130 SERVER COMPUTING DEVICE
121 VULNERABILITY TEST ENGINE
122 TEST RESULT REQUEST ENGINE
123 TEST RESULT RECEIPT ENGINE
124 VULNERABILITY DETECT ENGINE
129 DATA STORAGE
50 NETWORK
140A CLIENT COMPUTING DEVICE
140B CLIENT COMPUTING DEVICE
. . .
140N CLIENT COMPUTING DEVICE

FIG. 1

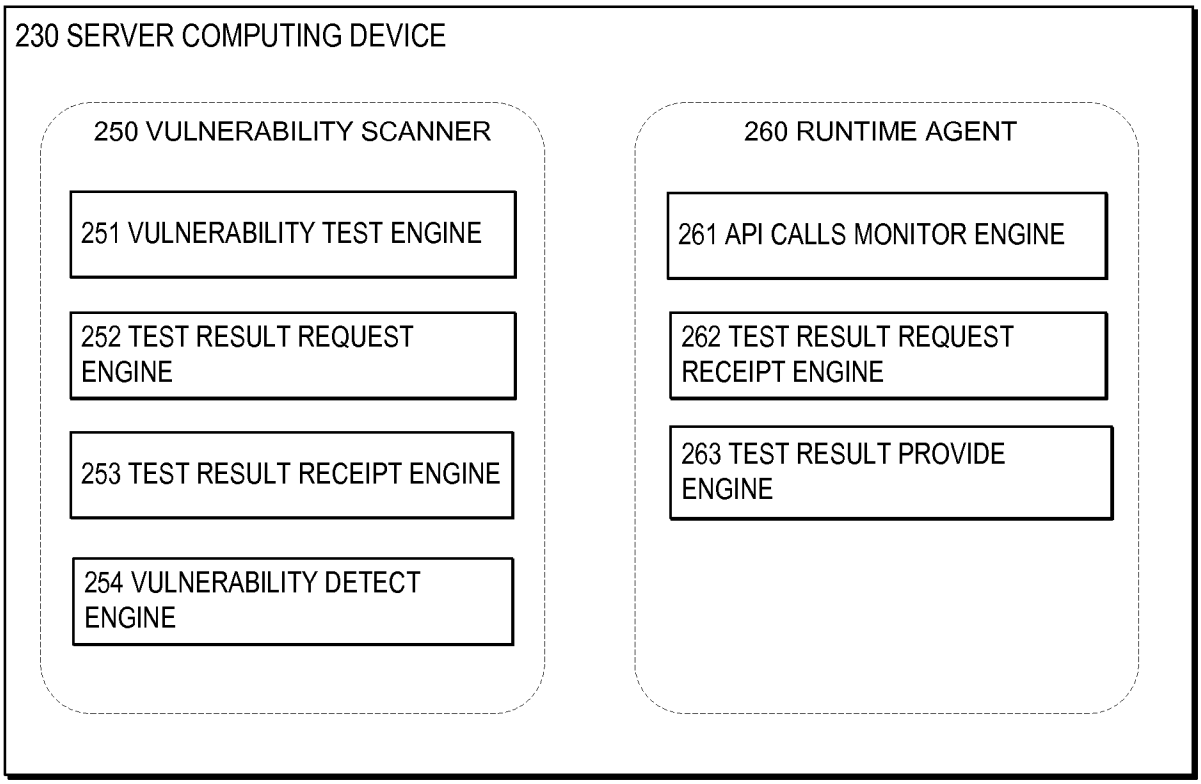

230 SERVER COMPUTING DEVICE
250 VULNERABILITY SCANNER
251 VULNERABILITY TEST ENGINE
252 TEST RESULT REQUEST ENGINE
253 TEST RESULT RECEIPT ENGINE
254 VULNERABILITY DETECT ENGINE
260 RUNTIME AGENT
261 API CALLS MONITOR ENGINE
262 TEST RESULT REQUEST RECEIPT ENGINE
263 TEST RESULT PROVIDE ENGINE

FIG. 2

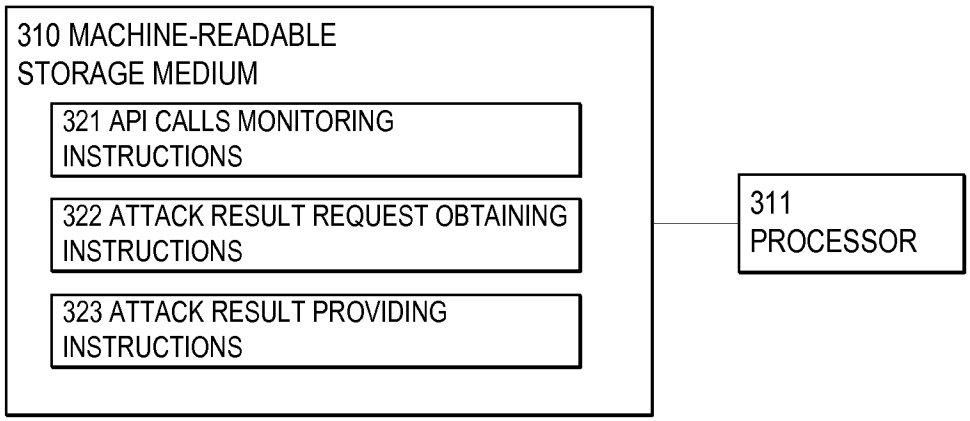

310 MACHINE-READABLE STORAGE MEDIUM
321 API CALLS MONITORING INSTRUCTIONS
322 ATTACK RESULT REQUEST OBTAINING INSTRUCTIONS
323 ATTACK RESULT PROVIDING INSTRUCTIONS
311 PROCESSOR

FIG. 3

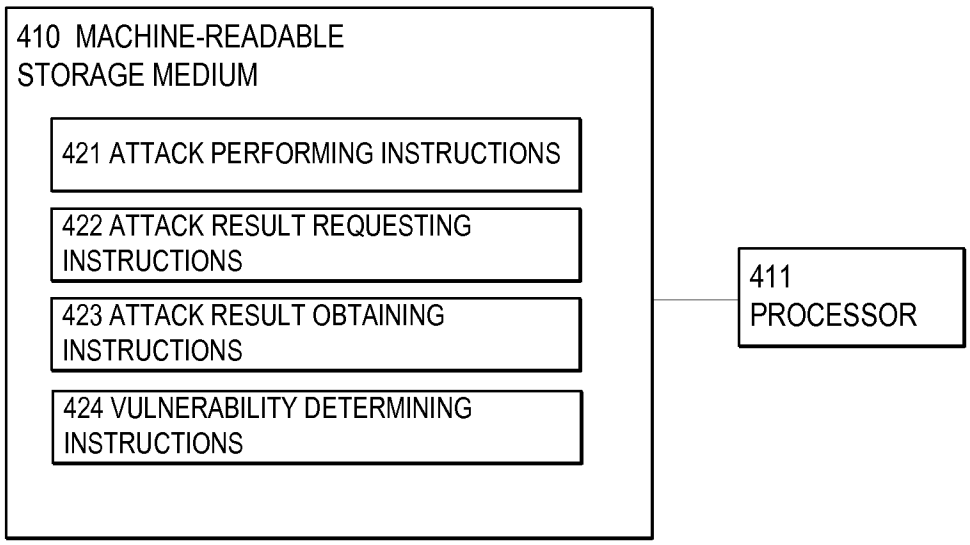

410 MACHINE-READABLE STORAGE MEDIUM
421 ATTACK PERFORMING INSTRUCTIONS
422 ATTACK RESULT REQUESTING INSTRUCTIONS
423 ATTACK RESULT OBTAINING INSTRUCTIONS
424 VULNERABILITY DETERMINING INSTRUCTIONS
411 PROCESSOR

FIG. 4

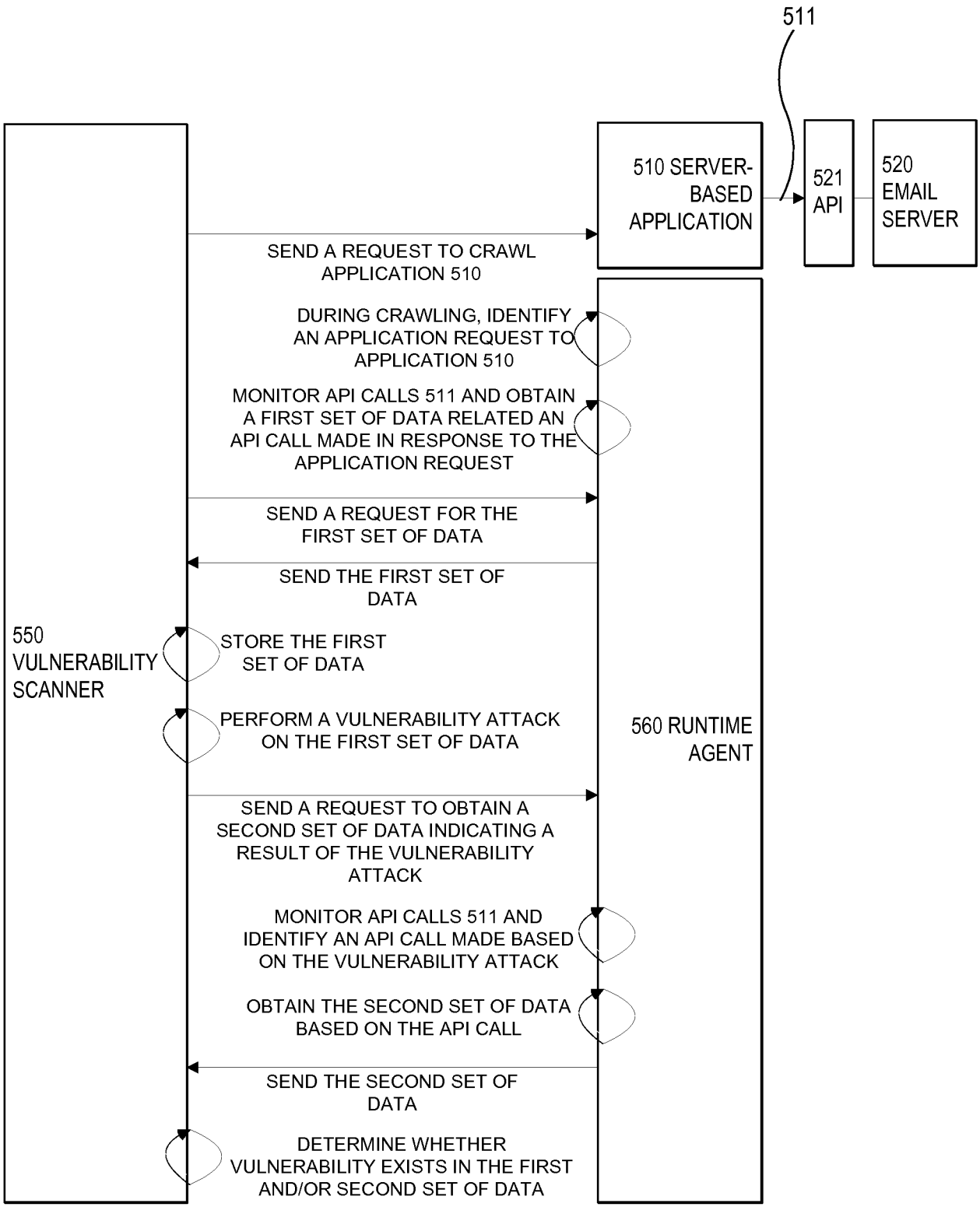

511
510 SERVER-BASED APPLICATION
521 API
520 EMAIL SERVER
SEND A REQUEST TO CRAWL APPLICATION 510
DURING CRAWLING, IDENTIFY AN APPLICATION REQUEST TO APPLICATION 510
MONITOR API CALLS 511 AND OBTAIN A FIRST SET OF DATA RELATED AN API CALL MADE IN RESPONSE TO THE APPLICATION REQUEST
SEND A REQUEST FOR THE FIRST SET OF DATA
SEND THE FIRST SET OF DATA
550 VULNERABILITY SCANNER
STORE THE FIRST SET OF DATA
PERFORM A VULNERABILITY ATTACK ON THE FIRST SET OF DATA
560 RUNTIME AGENT
SEND A REQUEST TO OBTAIN A SECOND SET OF DATA INDICATING A RESULT OF THE VULNERABILITY ATTACK
MONITOR API CALLS 511 AND IDENTIFY AN API CALL MADE BASED ON THE VULNERABILITY ATTACK
OBTAIN THE SECOND SET OF DATA BASED ON THE API CALL
SEND THE SECOND SET OF DATA
DETERMINE WHETHER VULNERABILITY EXISTS IN THE FIRST AND/OR SECOND SET OF DATA

FIG. 5

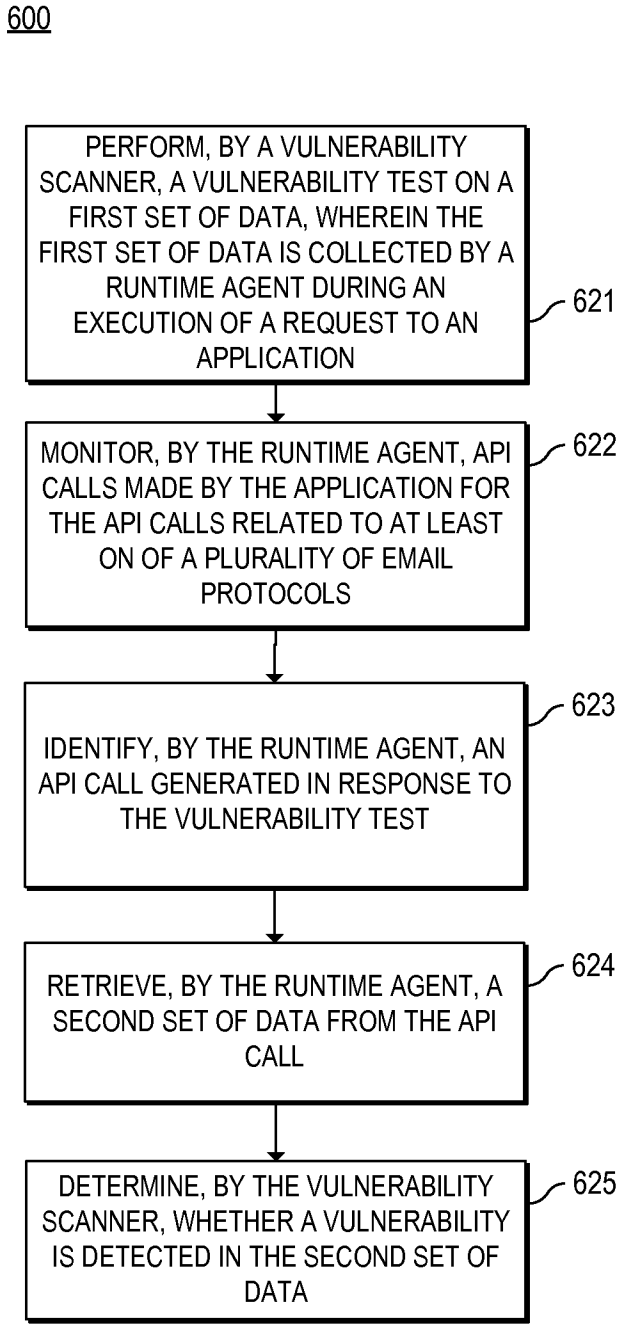

600
PERFORM, BY A VULNERABILITY SCANNER, A VULNERABILITY TEST ON A FIRST SET OF DATA, WHEREIN THE FIRST SET OF DATA IS COLLECTED BY A RUNTIME AGENT DURING AN EXECUTION OF A REQUEST TO AN APPLICATION
621
MONITOR, BY THE RUNTIME AGENT, API CALLS MADE BY THE APPLICATION FOR THE API CALLS RELATED TO AT LEAST ON OF A PLURALITY OF EMAIL PROTOCOLS
622
IDENTIFY, BY THE RUNTIME AGENT, AN API CALL GENERATED IN RESPONSE TO THE VULNERABILITY TEST
623
RETRIEVE, BY THE RUNTIME AGENT, A SECOND SET OF DATA FROM THE API CALL
624
DETERMINE, BY THE VULNERABILITY SCANNER, WHETHER A VULNERABILITY IS DETECTED IN THE SECOND SET OF DATA
625

FIG. 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2014/058144

A. CLASSIFICATION OF SUBJECT MATTER

H04L 12/26(2006.01)i, H04L 12/58(2006.01)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)
H04L 12/26; G06F 12/14; G06F 21/50; G06F 21/00; G06F 11/00; H04L 12/58

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: vulnerability, scanner, test, api, email, protocol

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 7325252 B2 (NELSON WALDO BUNKER, V et al.) 29 January 2008 See column 3, lines 55-64; column 14, lines 3-13; column 13, lines 55-56; column 17, lines 50-55; column 24, lines 32-33; claim 1, 9, 17; and figure 3.	1-15
A	KR 10-2014-0043081 A (HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.) 08 April 2014 See paragraphs [0015]-[0016]; claim 1; and figure 2.	1-15
A	US 8499354 B1 (SOURABH SATISH et al.) 30 July 2013 See column 6, lines 40-55; and figure 4.	1-15
A	US 2011-0307955 A1 (MARK KAPLAN et al.) 15 December 2011 See paragraphs [0055]-[0057]; claim 1; and figure 5.	1-15
A	US 2012-0304299 A1 (SAMIR GURUNATH KELEKAR) 29 November 2012 See paragraphs [0176]-[0178]; and figure 2.	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
09 June 2015 (09.06.2015)	**10 June 2015 (10.06.2015)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon Metropolitan City, 302-701, Republic of Korea
Facsimile No. +82-42-472-7140

Authorized officer
KIM, Seong Woo
Telephone No. +82-42-481-3348

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2014/058144

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 7325252 B2	29/01/2008	US 2003-0009696 A1	09/01/2003
		US 2003-0028803 A1	06/02/2003
		US 2003-0056116 A1	20/03/2003
		WO 2002-096013 A1	28/11/2002
KR 10-2014-0043081 A	08/04/2014	CN 103562923 A	05/02/2014
		EP 2715599 A1	09/04/2014
		EP 2715599 A4	04/03/2015
		JP 2014-517968 A	24/07/2014
		US 2014-0082739 A1	20/03/2014
		WO 2012-166120 A1	06/12/2012
US 8499354 B1	30/07/2013	None	
US 2011-0307955 A1	15/12/2011	US 2011-307951 A1	15/12/2011
		US 2011-307954 A1	15/12/2011
		US 2011-307956 A1	15/12/2011
		US 8881278 B2	04/11/2014
		US 8914879 B2	16/12/2014
		WO 2011-156592 A2	15/12/2011
		WO 2011-156652 A1	15/12/2011
		WO 2011-156679 A1	15/12/2011
		WO 2011-156754 A1	15/12/2011
US 2012-0304299 A1	29/11/2012	US 2005-0005169 A1	06/01/2005
		US 2013-0167240 A1	27/06/2013
		US 8127359 B2	28/02/2012
		US 8789193 B2	22/07/2014